Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Vascular damage detection through deep learning

Estudante: Ángel Rodríguez Martínez
Dirección: Manuel Francisco González Penedo
Víctor Manuel Mondéjar Guerra
Lucía Ramos García

A Coruña, November de 2019.

To my mother, my brother and Sara.

Acknowledgements

First of all I would like to thank my tutors Lucía, Víctor and Manuel for guiding me and helping me developing this project, their cheerfulness and passion for what they do was an invaluable source of motivation for me. Another big part of the knowledge and help I needed for this project came from my colleague Mateo. Finally I would like to thank the University of A Coruña for giving me a fantastic education and a lot of great moments to remember.

But none of this would have been possible without my family since they supported me every step of the way to where I am now. Especially my brother and my partner Sara who advised me to study computer science in the first place.

Abstract

HET-CAM (Hen's Egg Test - ChorioAllantoic Membrane) is a type of pharmaceutical analysis that measures the toxicity of a solution by administrating it to the chorioallantoic membrane of a fertilised egg. The membrane is used as an analogous tissue to that of the human eye in order to determine if a substance is suitable for human use. The process is recorded in video and analysed to chronologically locate three phases (haemorrhage, lysis and coagulation) that allow the classification of the toxic potential of the substance. HET-CAM is frequently used in the pharmaceutical industry to test substances, especially those destined to be used on the human eye. This approach has several advantages with respect to other methods like the Draize Test. It is easy to perform, affordable, and prevents unnecessary animal suffering. The main disadvantage is that the process is tedious and can be subjective as the beginning of each phase is manually marked by the person performing the analysis.

This project aims to use computer science techniques, specifically deep learning and image processing, to automatically analyse the videos and set a more reliable ground for the classification of the substances. These techniques applied to the HET-CAM videos allow the extraction of objective data that defines the processes taking place in the membrane. This data can then be studied by the system to make an initial classification of the substance which can then be used by experts to make a more informed decision.

Since there are no bibliographic precedents on solving this problem using artificial intelligence, part of the project will consist on the creation of a dataset that suits the needs of the chosen network architecture.

Resumen

HET-CAM (Hen's Egg Test - ChorioAllantoic Membrane) es un tipo de test clínico que sirve para medir la toxicidad de una sustancia al aplicarla a la membrana corioalantoidea de un huevo de gallina fecundado. La membrana sirve como análogo al tejido ocular humano a la hora de probar si un producto es apto para el consumo. El proceso es grabado en vídeo y analizado en función al orden cronológico de tres fases (hemorragia, lisis y coagulación) que permiten la clasificación del potencial irritante de la sustancia. HET-CAM se usa en la industria farmacéutica de manera frecuente, especialmente para fármacos destinados al ojo humano. Este método tiene varias ventajas con respecto a otros ensayos como el test de Draize. Es sencillo de realizar y previene sufrimiento animal innecesario. La principal desventaja es que el proceso es tedioso y subjetivo, ya que el inicio de las distintas fases es marcado de manera manual por una persona.

Este proyecto pretende usar técnicas de computación, especialmente deep learning y procesado de imagen, para analizar automáticamente los vídeos y sentar una base más sólida a la hora de clasificar las sustancias. Estas técnicas aplicadas a los vídeos HET-CAM permiten la extracción de datos objetivos que definan los procesos que están teniendo lugar en la membrana. Estos datos son estudiados por el sistema para hacer una clasificación inicial que los expertos pueden usar para tomar decisiones más informadas.

Dado que no hay antecedentes bibliográficos resolviendo este problema usando inteligencia artificial, parte del proyecto consistirá en la creación de un conjunto de datos que se amolde a las necesidades de la arquitectura de red escogida

Keywords:

- HET-CAM
- Deep learning
- Chorioallantoic membrane
- Bleed area
- Dataset
- Object recognition
- Video processing

Palabras chave:

- HET-CAM
- Aprendizaje profundo
- Membrana corioalantoidea
- Área de sangrado
- Set de datos
- Reconocimiento de objetos
- Procesado de vídeo

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Information Technologies (IT) are used in almost every part of the world to help humans solve complex problems in a fast and easy manner. One of the branches of IT is Artificial Intelligence (AI) which, as a broad description, is used to make a computer "think in a human-like fashion". This is very useful for solving problems that are not very well defined and therefore cannot be solved using an algorithmic approach, usually classification or regression problems.

1.1 Problem Description

In the pharmaceutical industry, several parameters need to be tested before releasing a drug to the market to ensure that it is suitable for human consumption. One of these parameters is the toxicity, which measures the potential ability of the drug to cause damage to human tissue. In the case of products destined to be used in the eyes, various tests exist, like the Draize Test[1] or the Hen's Egg Test - ChorioAllantoic Membrane (HET-CAM)[2]. The latter is the one that is going to be analysed in this study.

Hen's Egg Test - ChorioAllantoic Membrane (HET-CAM) consists in applying the drug to the chorioallantoic membrane of a fertilised chicken egg. This membrane, which is one of the inner membranes of the egg, is rich in blood vessels and behaves similarly to the human eye in regards to inflammatory processes. The egg needs to be fertilised for approximately 9 days, then, part of the shell is removed and the membrane is exposed. The drug is then applied to the membrane and the reactions taking place are observed by an expert to determine the three phases that are used to calculate the toxicity score (haemorrhage, lysis and coagulation). A greater amount of damage in a shorter period of time means that the product has greater toxicity. This procedure has several advantages with respect to other tests: it is affordable, and prevents animal suffering.

1.2 Motivation

The main drawback of HET-CAM is its low repeatability, mainly due to a subjective appreciation of the occurrence of reactions caused by the irritating substances, the differences among experts and the variability of the chorioallantoic membrane. Besides the subjectivity, the manual characterisation of the haemorrhage, lysis and coagulation timings is a tedious and time-consuming task. The nature of the HET-CAM procedure allows recording video footage of the process to analyse it at a later time. The motivation of this project is the automation of the assessment of that video footage in order to reduce the subjective character of the HET-CAM test allowing a reliable and repeatable extraction of objective data that supports the experts' decisions regarding the toxicity of the substance.

A previous study conducted by Gende [3] explores the use of computer vision techniques for extracting objective data from HET-CAM videos in terms of the amount of blood and blood vessels at each frame of the sequence. A big part of the project was dedicated to define the features needed to be analysed in order to get the wanted output. In this case, the areas of interest were the blood vessels and the ramifications where haemorrhages occur. To detect those areas using an algorithmic approach it was necessary to iteratively apply filters that accentuate the features and erase background and noise until valid outputs are obtained. The results extracted from this study shows that classical computer vision approaches allows to extract useful data from HET-CAM videos, however, in addition to the need of manually defining the relevant features to analyse, these techniques are affected by the variety of the data, the lighting conditions and are quite sensitive to parameter tuning.

In recent years, Deep Learning (DL) techniques have arisen as a promising alternative to address object detection problems given its ability to recognise complex patterns from raw input data, learning proper hierarchical representations of the underlying information at different levels. In the context of HET-CAM test, the project would be considered as an object recognition problem where the haemorrhage areas in each frame need to be detected.

Therefore, in this work, it is proposed the use of DL techniques for extracting objective information from HET-CAM videos. By solving this problem using DL a more robust system can be created, being less vulnerable to noise, colour or illumination changes, and supporting multiple resolutions. Making the program support all kinds of videos will also promote the use of HET-CAM in otherwise reluctant laboratories.

1.3 Objectives

The main goal of this project is to develop an artificial intelligence system that takes the videos and extracts data regarding the irritation happening on the membrane (in the form of graphs

that show the evolution of bleeding areas) and displays it in a way that helps experts make a decision on the toxicity of the drug. To accomplish this objective, the system will look at the videos as an image recognition problem. The system will analyse the videos frame by frame detecting each area of bleeding, then the output will be constructed using this information. The complete problem was divided into several sub-objectives that need to be achieved.

- **Create a dataset:** A dataset is the collection of tagged data used to train and validate AI systems. Since there is not bibliography in using AI systems to analyse HET-CAM videos, a custom dataset needs to be created from the videos that are available. A program will be developed to aid in the transformation of the raw data into an annotated dataset.
- **Train several network configurations:** Several parameters and configuration options can be changed in AI systems. The consequences of changing this factors will be studied in order to achieve the most appropriate configuration. Some of the parameters that are going to be considered are: variations on the learning rate, use of pretrained models or data augmentation.
- **Compare the results with experts' annotations:** Finally, once the appropriate system configuration is chosen, the data extracted from the videos will be compared with the annotations made by the experts to see how they are correlated.

1.4 Outline

This document is split into 6 different chapters to help the reader understand the problem, the methodology followed and the outcomes. The planning and time scheduling of the process are detailed in Appendix A.

- **Chapter 1: Introduction** This chapter provides an introduction to the problem, detailing its description, the motivation behind the work, and the objectives that need to be accomplished.
- **Chapter 2: Domain description** This chapter contains an explanation of the domain of the problem, including a more detailed description of the HET-CAM test, as well as the information needed to implement the artificial intelligence network.
- **Chapter 3: Methodology** This chapter constitutes the main body of the document and presents the proposed methodology to address the problem. It encapsulates all the proceedings of creating the dataset, from the gathering of HET-CAM videos to the concrete method to annotate them, as well as the development of a marking tool to

help with the annotation procedure. Then, the bleeding detection module is described, including the extensive process for the parameter adjustment. Finally, the bleeding evolution module that allows generating informative graphs for the HET-CAM videos is presented.

- **Chapter 4: Software development** This chapter provides a detailed view of the design and implementation of the different software tools developed and used throughout the project.
- **Chapter 5: Results** This chapter presents the results provided by the proposed methodology regarding the bleeding detection and the bleeding evolution throughout the HET-CAM videos, assessing the distinction between negative and positive videos, as well as the relation with the experts' timings. Additionally, the extracted results are discussed, highlighting the potential and constraints of the methodology.
- **Chapter 6: Conclusions and future work** In this chapter, the results of the project are analysed and the possible lines of work and enhancements to the program are discussed.

Chapter 2

Domain description

This project consist on the implementation of an IT system, specifically Deep Learning, to be used in the pharmaceutical industry to serve as support for the experts when analysing the toxicity of drugs via the HET-CAM test. In this chapter, all preliminary knowledge needed on HET-CAM and Deep Learnig will be presented.

2.1 HET-CAM

The toxicity of a drug is the potential it has to harm live tissue. Toxicity testing is a part of the preclinical testing phase in the development cycle of a drug and it must be done before studies on humans can begin in order to avoid unnecessary damage. HET-CAM is the acronym for Hen's Egg Test - ChorioAllantoic Membrane and it is a procedure to where the membrane of a hen's egg is used to test the drug because of its similar vascular structure to the human eye. The drug is applied and the reactions taking place in the membrane are observed to determine the toxicity score of the substance. HET-CAM was found to return similar results to the Draize test [4] with the advantage that it is not performed over live animals.

To start the procedure, the egg needs to be fertilised for 9 days. Then the egg is placed on a stand and an opening is cut in the outer shell using a saw. Then the outer most membrane is retired, leaving the CAM visible. Now the egg is ready for the test. It is placed below the recording device of choice and the solution to be tested is applied to the membrane using a pipette. The reactions taking place in the membrane are recorded for 5 minutes, then the test is considered complete. The whole procedure is summarised in figure 2.1 (page 6).

At some point in these five minutes, if the substance is irritant, a haemorrhage occurs, causing blood to spill from the blood vessels and form round haemorrhage spots. There are three different stages of interest: haemorrhage, which corresponds to the spillage of blood out of the capillaries; lysis, which is the point where the blood vessels reach their maximum size and burst; and coagulation, which is the reduction on the haemorrhage rate due to the

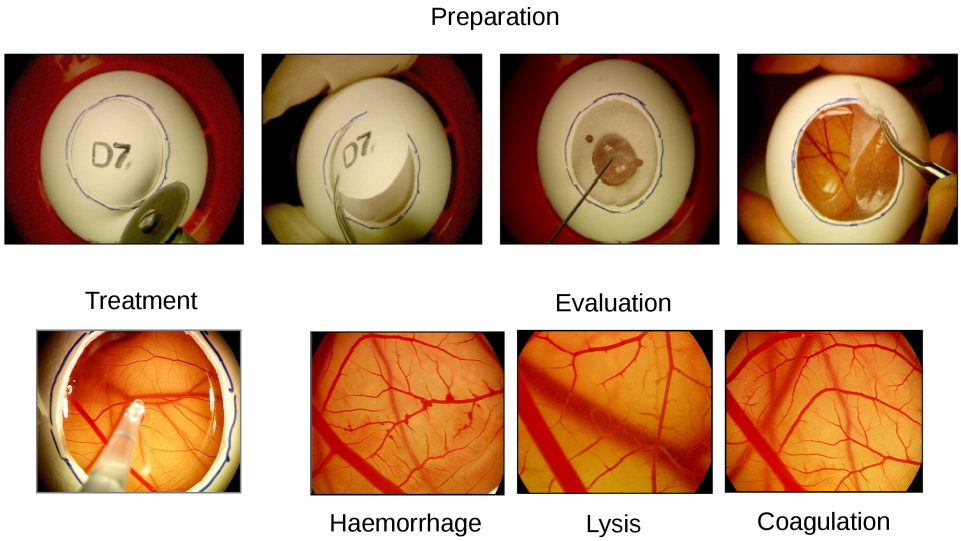

Figure 2.1: The steps of the HET-CAM test

solidification of the blood outside blood vessels. The timing of these three stages determines how irritant a drug is. A visual differentiation between stages can be observed in figure 2.2 (page 7)

The toxicity of the drug is represented by the so-called Irritation Score (IS). It is calculated using the time at which these three stages occur. The following formula relates the haemorrhage time T_H, lysis time T_L and coagulation time T_C with the irritation score [4].

$$IS = \left(\frac{301 - T_H}{300} \cdot 5\right) + \left(\frac{301 - T_L}{300} \cdot 7\right) + \left(\frac{301 - T_C}{300} \cdot 9\right)$$

This process has the advantage that it is an *in vitro* test, whereas the Draize test is *in vivo*. This means that the procedure is not performed over a living animal, thereby preventing unnecessary suffering to the creature. Some other advantages lie in the process being affordable and easy to perform. However, this test is affected by low repeatability, mainly due to a subjective appreciation of the reactions related to irritating substances, the differences among experts, and the variability of the chorioallantoic membrane. Furthermore, manual annotation of the stages of the process is a tedious and time-consuming task. Therefore, automated processes for the assessment of the membrane evolution would reduce the subjective character of the test, providing valuable, objective information to the experts to support the decision making processes.

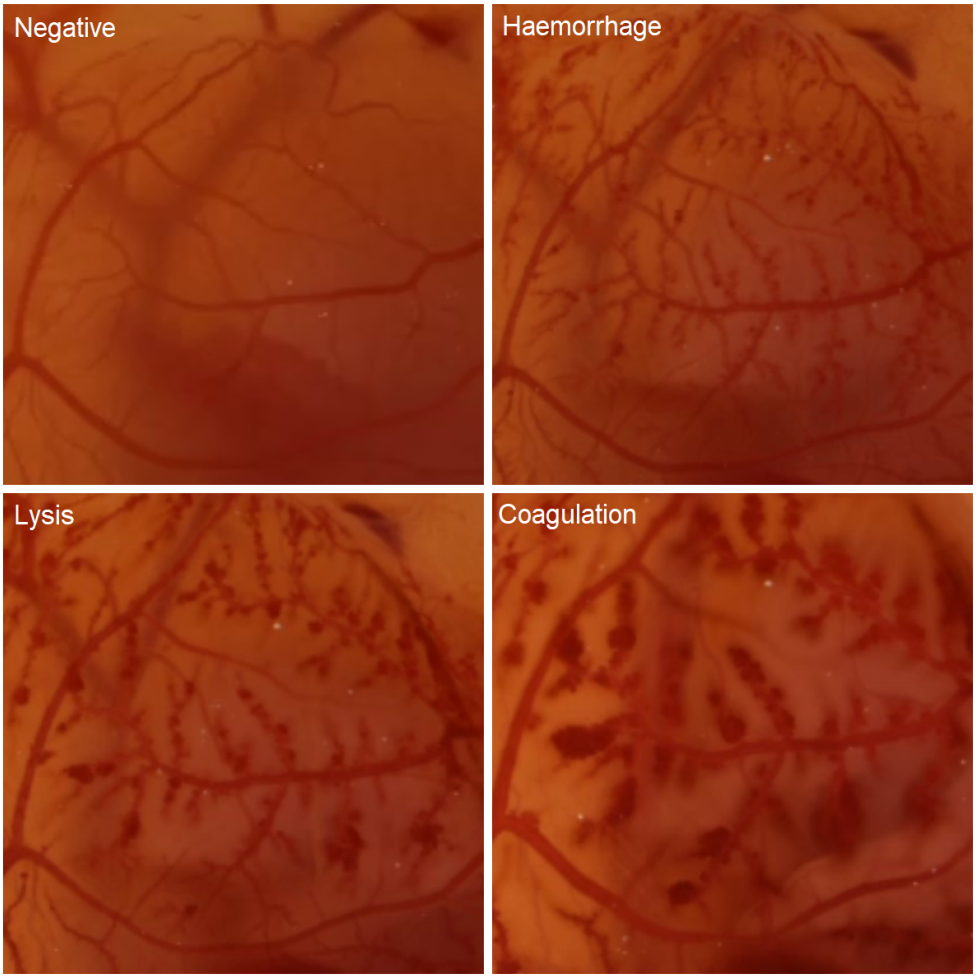

Figure 2.2: Four different frames corresponding to the stages of the irritation process given by an expert

2.2 Deep learning

During the last decade, Deep Learning has demonstrated to be an excellent technique in the area of artificial intelligence, solving different problems, and even surpassing humans in some cases [5, 6]. In addition, DL yields good results in diverse areas like image recognition [7], medical imaging [8], or speech recognition [9].

Classic machine learning approaches require designing feature extraction methods in order to generate a descriptor (i.e., a feature vector that emphasises the pattern to detect and that is usually more compact than the raw data). The descriptor feeds a learning algorithm that performs a specific task, such as a classification or a regression of the input data. In contrast to such an approach, the main advantage of DL methods is the capacity to recognise com-

plex patterns directly from the raw data, learning proper hierarchical representations of the underlying information at different levels. Therefore, DL methods are able to perform both tasks (the representation of the data and the classification) at the same time just by feeding the network with raw data [5].

This feature of DL makes it a good candidate for object recognition problems. Being able to train the network in a set of images whose desired output is known can save a lot of time and return better results in comparison to trying to detect the objects via computer vision techniques. A well known object detection algorithm based on artificial intelligence, designed by Girshick et al.[10], mixes Convolutional Neural Networks (CNN) with a concept called region proposals. The performance of this system showed a large increase in performance compared to other methods. Since then, object recognition using CNNs has seen lots of developments [11][12][13][14], which points to this type of system being the appropriate one for the problem at hand.

The problem with object detectors like R-CNN [10] is that the require a good amount of pre-processing of the images and use other complex algorithms to filter the output. Optimising a model like this to a specific problem requires a lot of fine tuning, since the different parts of the model are optimised separately. An alternative method of object detection using deep learning is the You Only Look Once (YOLO) model [13]. This model encapsulates the whole procedure in a single CNN, taking as inputs the raw images and giving bounding boxes of objects as outputs.

In the specific context of this project, the YOLO architecture has been previously used to address the detection of structures with morphological characteristics similar to the haemorrhages produced in HET-CAM videos, such as the detection of carcinogenic masses in mammography images [15] or the detection of lung nodules in computed tomography scans [16]. The results achieved in these proposals serve as a precedent to promote the use of the YOLO architecture for detecting the bleeding areas in HET-CAM videos.

2.2.1 You Only Look Once (YOLO)

The YOLO architecture consists of a single neural network that predicts bounding boxes and class probabilities directly from full images in one evaluation. First, the input image is rescaled to the working image size of the network. Then a grid that divides the image into chunks is created. If an object's centre falls into a chunk, that cell is responsible for returning the detection of the object. Each object detection is also accompanied by a confidence score that represents the probability of that object being a correct detection. Each cell also computes the probability of an object of each of the classes being presented in it. Finally, the model computes the relation between bounding boxes and class probabilities in order to assign a class to each box. The boxes detected, with their centre coordinates, height, width, class and

confidence are the output of the network. Finally, non maximum suppression is applied to the outputs to erase detections with less than a confidence threshold. The process is graphically represented in figure 2.3 (page 9).

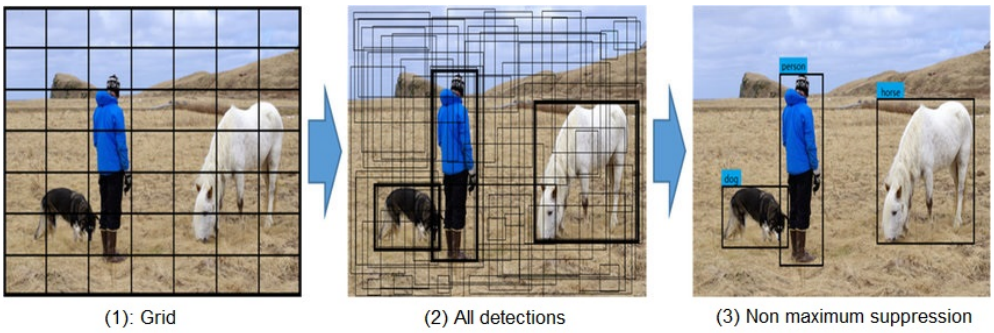

Figure 2.3: Simple explanation of the YOLO object recognition system

The architecture of the neural network (Figure 2.4, page 9) is comprised of 24 convolutional layers followed by 2 fully connected layers. The final layer of the network calculates the bounding boxes as well as the class probabilities. Bounding boxes are normalised between 0 and 1.

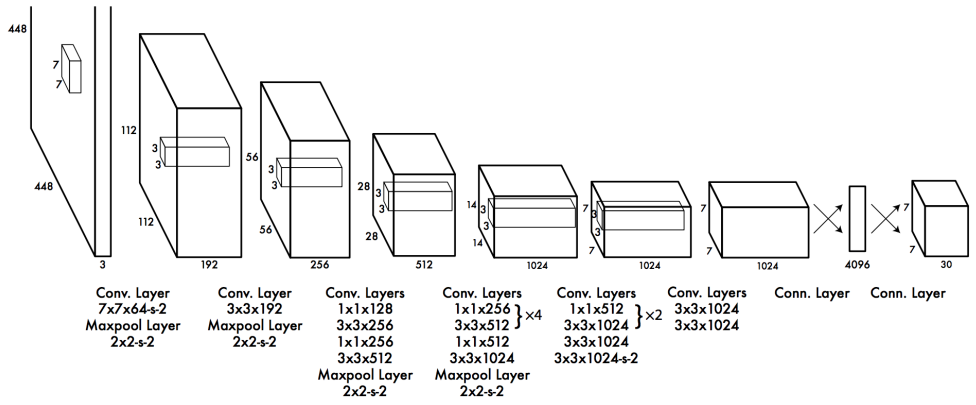

Figure 2.4: YOLO network architecture

The loss function that is optimised during training is composed of a series of factors that are then summed to get the total loss.

- **Bounding box location and dimensions:** The mean squared error is used to calculate the difference between each detection and its ground truth counterpart.
- **Class prediction:** Binary cross entropy is calculated to refer to the amount of objects of one class detected as other classes.

- **Confidence loss:** It also uses the binary cross entropy formula. This term is calculated for each grid cell if it is responsible for a detection and it computes the class confidence of that cell compared with the ground truth class value for that cell.

The confidence loss can be driven towards 0 by the cells that do not count with any object. Taking this into account, two factors are introduced in the formula to balance the weight confidence loss and bounding box loss. For a more detailed view on the loss formula and more specific details of the model, the reader can refer to reference [13].

Another feature of the network is that it looks at the image as a whole, instead of trying to find objects locally like other algorithms like deformable part models that use a sliding window. This allows the network to make less false positive mistakes over the background of the image,

Chapter 3
Methodology

The steps taken to solve the problem will be specified in this chapter. First, an outline of each step will be presented, lightly detailing what needs to be done. The following sections will tackle the concrete analysis, implementation and result of each step.

3.1 Methodology outline

In order to analyse the videos and automatically gather valuable information from them, an application will be developed that uses a deep learning network to find the areas of interest. This structure will be used to locate the areas of bleeding in each frame to later on print graphs that represent the evolution of the bleeding area with respect to the time in the videos. The proposed methodology to analyse the HET-CAM videos is the following, as it can be seen in figure 3.1 (page 12):

- **Capturing video frames:** in fixed size batches to be passed to the deep learning system for their analysis. Some pre-processing might be needed, such as re-scaling each frame to a fixed resolution or skipping a number of frames between samples.
- **Analysing the frames using object detection via deep learning:** The data from the previous step is passed to the artificial intelligence system to extract valuable information from each frame, i.e. the areas detected as bleeding and their position and size.
- **Building informative graphs:** using the information gathered by the network for the experts to make decisions on the toxicity of the substance.

The key part of the methodology is the deep learning network. The You Only Look Once (YOLO) architecture[14] was chosen to take care of object detection for its performance and, even though execution times are not very important in this problem, its ability to process

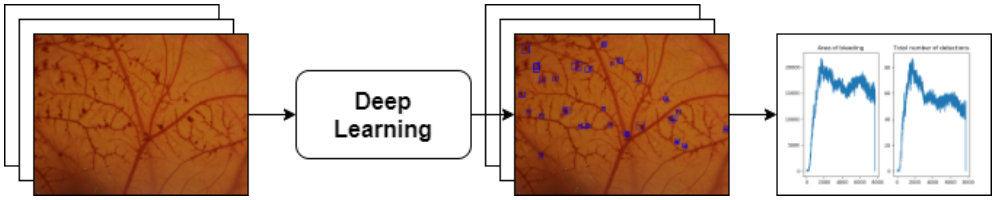

Figure 3.1: Methodology outline diagram

near real time video. This second characteristic is not crucial for the concrete implementation detailed in this document, but analysing a real time video feed (instead of pre-recorded videos) can be a feasible point of extension for future work in the program. YOLO was also chosen for its low number of background errors compared to other algorithms [13], since the area occupied by the objects is going to be measured, detecting part of the background as a haemorrhage is not desirable.

The network needs to be trained properly in order to be as robust and generalist as it can be. For the training process, the following steps need to be accomplished

- **Build a dataset:** since there is no bibliography on applying artificial intelligence to HET-CAM, there are no datasets available either. Building a dataset following the YOLO specification from the available videos will be the first step of the project.
- **Train several network configurations:** Various parameters need to be tested in order to obtain the network that best fits the problem. Experiments will be conducted altering these parameters and performance metrics will be recorded in order to pick the most appropriate model.
- **Choose the most appropriate configuration:** The results of the training (confusion matrices and other metrics) will be studied to pick the configuration that best fits the problem.

Finally, an application that uses the trained model to show the evolution of the bleeding areas of videos will be developed. The output of the program will then be compared to the times given by the experts in order to find if the information extracted by the network is useful in helping experts decide on the toxicity of a drug.

3.2 Dataset

Given the nature of the DL methods that extract complex hierarchical abstractions from the data used throughout the training stages, the effectiveness and capability of generalisation is bounded by the quantity and quality of the used dataset. Therefore, an extensive amount

of tagged data is needed so the system can learn which features are important in a certain context. Some examples of datasets are NMIST[17] which contains thousand of pictures of handwritten numbers, or ImageNet[18] which is a database of images of objects of all sorts.

In the context of this project, a dataset comprised of the different bleeding areas in HET-CAM videos is needed. No dataset tackling this concrete context was found so the creation of one was required. This section's focus is to specify everything concerning the dataset: the source of the data, how it was annotated, and the limitations and problems that might be related with it.

3.2.1 Video data available

The videos used to build the dataset were recorded by experts of the Pharmacy Faculty of the University of Santiago de Compostela (USC) in December 2017. In the videos provided, both positive and negative test are performed on the same egg, one after the other. The negative control corresponds to a solution of sodium chloride (NaCl) and the positive corresponds to a solution of sodium hydroxide(NaOH). In the negative test no change in the membrane is seen during the whole length of the video. However, in the positive test, little accumulations of blood begin to appear around the blood vessels as the time passes, as seen in figure 3.2 (page 13). The dataset will be comprised of images corresponding to different frames of each video, where the centre and size of the haemorrhage spots will be annotated.

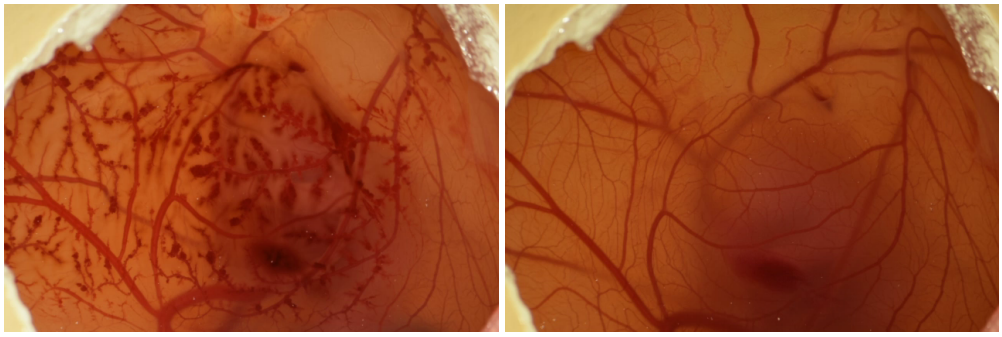

Figure 3.2: Positive and negative frames

The data available consists in 18 HET-CAM videos corresponding to 7 different test eggs with a negative and a positive test in each egg. Most of the videos are around 5 minutes in length and there are recordings in three different resolutions across two aspect ratios (16:9, and 4:3). The specification of the data can be seen in table 3.1.

From the 7 positive videos available, only 5 were annotated by the experts according to the haemorrhage, lysis and coagulation times. These will be the videos used to conform the dataset. It was found that consecutive frames are very similar to each other, so the different images gathered from each video will be separated in regular intervals of 240 frames. This

Egg	Resolution	Quality	Filename	Solution
1	1920x1080, 50p	High	DSC_1087	Nothing
1	1920x1080, 50p	High	DSC_1088	NaCl
1	1920x1080, 50p	High	DSC_1089	NaOH 0.1 N
2	1920x1080, 50p	High	DSC_1090	Nothing
2	1920x1080, 50p	High	DSC_1091	NaCl
2	1920x1080, 50p	High	DSC_1092	NaOH 0.1 N
3	1920x1080, 50p	High	DSC_1093	Nothing
3	1920x1080, 50p	High	DSC_1094	NaOH 0.1 N
4	1280x720, 50p	High	DSC_1096	Nothing
4	1280x720, 50p	High	DSC_1097	NaCl
4	1280x720, 50p	High	DSC_1098	NaOH 0.1 N
5	640x424, 25p	Low	DSC_1099	Nothing
5	640x424, 25p	Low	DSC_1104	NaOH 0.1 N
6	640x424, 25p	Low	DSC_1105	Nothing
6	640x424, 25p	Low	DSC_1106	NaCl
6	640x424, 25p	Low	DSC_1107	NaOH 0.1 N
7	640x424, 25p	Low	DSC_1108	Nothing
7	640x424, 25p	Low	DSC_1109	NaOH 0.1 N

Table 3.1: Source data specification

measure was taken in order not to include a lot of copies of the same haemorrhage point, since they evolve slowly.

For the negative tests, a set of 10 representative frames were selected in order to cover the contrast and illumination variability of the HET-CAM videos. Since negative controls do not cause bleeding areas, manual marking of these frames is not needed.

3.2.2 Data tag specification

The different haemorrhages that comprises the dataset were annotated by a bounding box, which is delimited by the centre point (x, y), its width and height. Each image in the dataset must be accompanied by a *.txt* file with the same name as the image where each line represents the bounding box of an object. Table 3.2 describes the structure of each line in the file.

Class index	Centre_X	Centre_Y	Width	Height

Table 3.2: A data tag sample

The class index is a number that represents the type of object that is being annotated. In our case there is only one class, a haemorrhage area, so the class index of all of the tagged objects will be 0.

The *Centre_X* and *Centre_Y* specify the position of the centre of the object and *Width* and *Height* represent its dimensions, all of them bounded between 0 and 1 with respect to the size of the image.

3.2.3 Data selection

A series of programs were developed to carry out the extraction of images from the videos and the later annotation of each of the selected frames. The development process of these tools is detailed in chapter 4. But not all the frames in the videos are going to be annotated and included in the dataset. This sections objective is to explain the process that was followed to go from the raw video data, to the series of images that are going to be annotated in terms of their haemorrhage spots.

As it was stated in section 3.2.1, from the 7 positive videos available only 5 were annotated by the experts in terms of the haemorrhage, lysis and coagulation times. These where the 5 videos used to conform the dataset.

Using the scripts mentioned in section 4.1, image samples were taken from each video. The haemorrhage growth rate is slow at the beginning and is almost none at the end of the videos, therefore the amount of frames skipped between samples was set to 240, in order not to include a lot of duplicates of each haemorrhage point. It was observed that around the one minute mark, the variation of haemorrhage spots (even 240 frames apart) was almost negligible. There is a point in each video were the bleeding stops evolving and just becomes distorted and blurry, when different areas begin to merge and form big clots that occupy a large portion the picture. At this point the images were no longer included in the dataset. Therefore, only the first 10 images from each video were selected.

From the images selected, the ones that had a 16:9 aspect ratio were converted to 4:3. To avoid having duplicated haemorrhage areas in the dataset due to the overlapping region of the sub-images, just the most representative side was chosen to form part of the dataset.

3.2.4 Marking process

Once the frames of interest were extracted from the HET-CAM videos, the bleeding areas of each of these frames were manually marked using the Image Marker program detailed in section 4.2. Some rules were defined in order to annotate the dataset in a coherent manner. The first step was defining what a haemorrhage point is and how to mark it. Haemorrhage points appear on the sides and terminations of blood vessels when these break and let blood out. The mark assigned to each bleeding area is slightly oversized in order to have a full representation of the haemorrhage area as well as the contrast between the haemorrhage's boundaries and the background, as it can be seen in figure 3.3 (page 16).

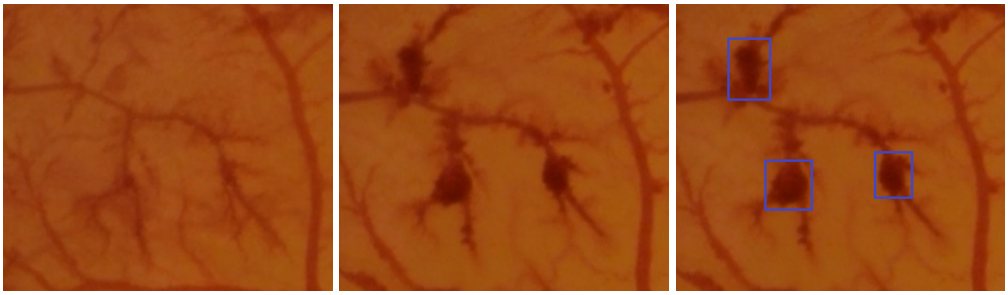

Figure 3.3: Healthy blood vessel - Haemorrhage - Marked haemorrhage

With this in mind, the marking process began with the higher resolution videos. An exhaustive search of the bleeding areas was performed, accumulating over 100 marks per frame (excluding the first frames in the videos, where the haemorrhage only recently started). Just in the first video (DSC-1089), which was HD converted to 4:3 aspect ratio (1440x1080 resolution), 1140 bleeding spots were marked. However, when proceeding to mark the lower resolution videos, like DSC-1104, it was found that the smaller bleeding areas that were marked in the previous (higher quality) video were barely recognisable, being regions of less than 100 $pixel^2$ ($10x10px$ area). In order to preserve the marking process coherent between the videos, a low bound was set on the size of the area of the regions to be marked, that is, just including the marks that were clearly visible without zooming in the picture but were still recognisable (not a $10x10px$ blur) while zoomed. As the appearance probability and growth rate of the bleeding area seems to be uniform throughout the picture, this criteria does not harm the validity of the data gathered, as ignoring the smaller areas will mean that they will just appear later on the analysing process, when they reach the size threshold. Not only that, but knowing this limitation on the dataset can allow the program to be calibrated accordingly to contemplate the possibility of non-detected, small size haemorrhages. The difference between exhaustive marking and annotating following this new rule can be observed in figure 3.4 (page 16).

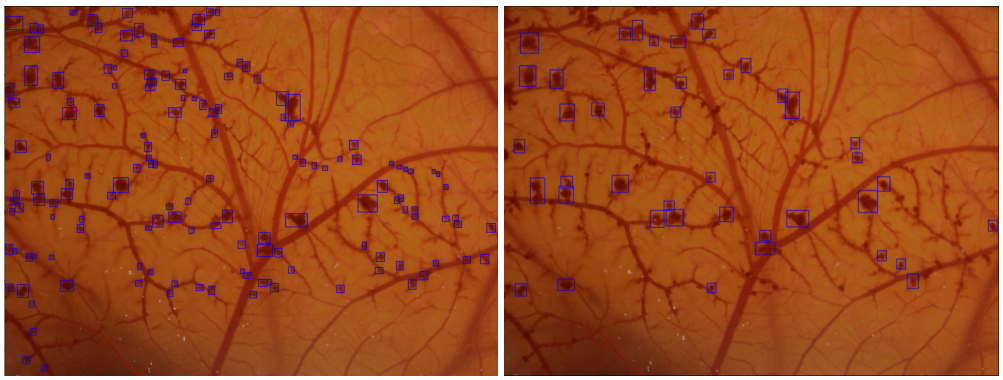

Figure 3.4: Exhaustive marking (154 marks) vs. ignoring too small haemorrhages (42 marks)

Another type of haemorrhage that was excluded from the dataset occurs towards the end of the videos, when the blood expands over a large area and forms big clusters of blurry bleeding areas. Due to the constraint of having to mark rectangular areas and with the purpose of not including a lot background in the samples, these blurred areas were not included in the dataset. As it is visible in figure 3.5 (page 17) only the well defined haemorrhages with high-contrast borders were marked.

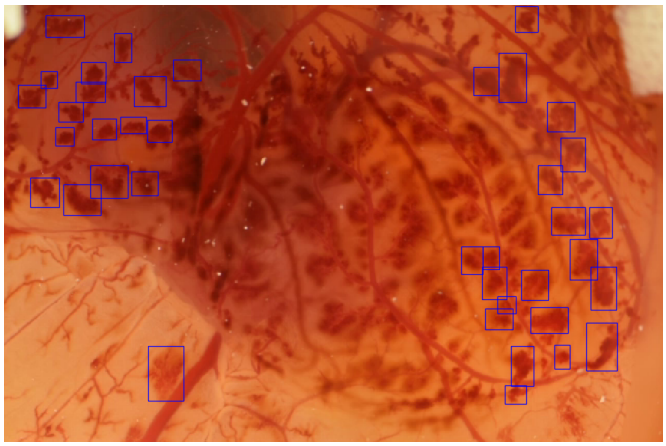

Figure 3.5: Blurry haemorrhages that were not annotated

Once the marking criteria were defined, the higher resolution images were re-marked and the procedure continued until all the dataset was annotated. Table 3.3 summarises the annotated videos including the number of frames per video as well as the total number of haemorrhage areas tagged across all the frames.

Video	Frames marked	Total number of areas
DSC_1089	10	338
DSC_1098	10	211
DSC_1104	10	278
DSC_1107	10	431
DSC_1109	6	338
Negative_Videos	10	0
Total:	56	1596

Table 3.3: Final state of the dataset

3.3 Bleeding detection module

The first module of the proposed methodology is focused on analysing the HET-CAM frames in order to detect the bleeding areas. This module is based in deep learning techniques, in

particular, the YOLO architecture, as mentioned in section 3.1.

A base implementation of the network is granted by Linder-Norén [19] and it uses the Python language, with Pythorch and TensorFlow as its main libraries. This generic implementation will be tuned up in a number of different ways to best fit the problem at hand. That is, an iterative programming methodology (Figure 3.6, page 18) will be followed, which each iteration corresponding to an experiment over the parameters of the network. This section explains every iteration performed to find an adequate configuration of the network.

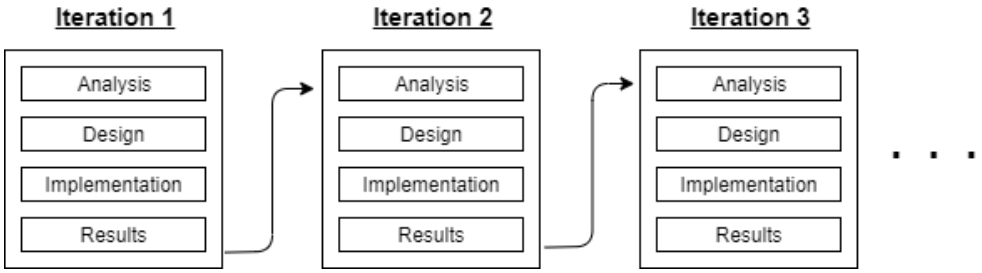

Figure 3.6: Diagram of an iterative programming methodology

3.3.1 YOLO implementation

As a starting point, the implementation developed by Linder-Norén will be used [19]. It is developed in Python taking advantage of the popular artificial intelligence libraries Pytorch and Tensorflow. The program is compatible with CUDA, which is integrated in Pytorch, so the process can be accelerated by using an Nvidia GPU. This implementation was chosen because it is easy to use and modify, and also allows training over a custom dataset.

This program consists of several scripts that manage the different stages of the process: from training and logging the results to using a trained model to process a set of images. Prior to beginning to modify the program, a rough understanding of its components and its data flow was required. This was achieved with help from the examples and the documentation provided by the developer. Some of the most important scripts are detailed ahead.

- *train.py:* Allows the network to be trained by taking several parameters like: the dataset, the number of epochs, the intersection over union (IoU) threshold to consider that a detected box corresponds to the target one...This script performs all the train and validation process, saving performance metrics using Tensorboard, and also storing the weights of the network in regular intervals called checkpoints. This behaviour allows the user to study the performance of the network as epochs go by, and choose the appropriate epoch from which to take the weights that will be used in the final program.

- *detect.py:* It serves as an example of how to use a trained model. It takes the trained

weights and a folder of images and passes them through the network. It then prints the bounding boxes of the objects detected on each image and saves them in a different folder.

- *test.py:* It is used to validate the network during the training process but it can be used on its own to validate any model over the validation set.

One important enhancement included was validating the model over the set of negative (no haemorrhage) pictures and count the number of objects detected to ensure a good differentiation between positive and negative videos.

3.3.2 Performance metrics

The performance of the deep learning module is evaluated by comparing the detections provided by the network with the target areas annotated in the dataset. The confusion matrix is a set of values commonly used in machine learning to evaluate the performance of a system. Since the model being analysed only has one class (haemorrhage) the matrix will have two rows corresponding to the ground truth values (haemorrhage or not haemorrhage) and two columns corresponding to the values detected by the model. Therefore, there are four cells in the matrix that correspond to the four different types of detections, as it is stated in table 3.4.

Ground Truth \ Detected	Haemorrhage	No haemorrhage
Haemorrhage	True Positive	False Negative
No haemorrhage	False Positive	True Negative

Table 3.4: Confusion matrix values

- **True positive:** A haemorrhage spot that is tagged in the dataset and then detected by the network. The model allows for slight discrepancies with the alignment and size of the detected object with respect to the ground truth. An intersection over union (IoU) greater than 0.5 between the ground truth and the detected object is needed to consider it as a true positive.

- **False positive:** An area with no haemorrhage that is detected as a haemorrhage by the system. Since the dataset is not perfectly annotated, the number of detections considered false positives will be higher than the reality. This is caused by not all haemorrhages being tagged, therefore some haemorrhages will be detected by the system that are not annotated in the ground truth. These spots will count as false positives even though they are true positives.

- **True negative:** No haemorrhage is tagged by the ground truth or detected by the model. This value is related to the amount of background that is indeed detected as background.

- **False negative:** A haemorrhage spot present in the ground truth but not detected by the network. Since the dataset was annotated with only the most representative haemorrhage spots, it is imperative to minimise the number of false negatives, ensuring that the system can detect most of the annotated objects in the ground truth.

The confusion matrix is a representation of the raw number of detected objects with respect to the ground truth. In order to have a more representative system to evaluate the network, some calculations will be performed using the confusion matrix's numbers to transform them into several ratios.

- **Recall:** or true positive rate, represents the amount of objects present in the ground truth that were detected by the system. A higher recall will be prioritised over other metrics because, as said before, the dataset is only annotated with the minimum set haemorrhages that need to be detected. A high recall means that more of the ground truth's objects were detected by the network, whereas a low recall means that a high number of targets were not found.

$$recall = \frac{true\_positives}{true\_positives + false\_negatives}$$

- **Precision:** is a measure of the number of false positives out of the total number of detections. It depicts how many of the detections were actually true positives. The more detections are true positives will yield a higher precision, but if the number of false positives is high, this metric will decrease.

$$precision = \frac{true\_positives}{true\_positives + false\_positives}$$

- **F1 Score:** represents the relation between recall and precision to give one unique value that represents the performance of the network.

$$F_1 = 2 \cdot \frac{recall \cdot precision}{recall + precision}$$

It is usual to calculate these performance metrics for every individual image and then compute the mean for the whole dataset. However, since positive tests usually implies an increase in the number of bleeding areas over time, the dataset has a lot of variability regarding

the number of target objects in each image, ranging from 10 to over 50 objects per image. The weight that every single detection has the performance of an image can vary significantly. For example, in an image with 10 target objects, not detecting one of them decreases the recall by 10% but missing one detection in an image with 100 target objects decreases the recall by just 1%. In order to ensure that each target object has the same weight, the performance metrics were computed over the confusion matrix of the whole dataset.

One of the functions that need to be achieved by the final program is distinguish between positive and negative videos, hence the number of detections (false positives) in negative videos is also analysed. Ideally, the system would detect 0 false positives in every frame that corresponds to a negative video, but some areas (that are not bleeding) can be recognised as haemorrhages even by the human eye. Some of these areas correspond to healthy blood vessels or parts of the embryo in the background. The target of this value is to be as low as possible.

3.3.3 Cross-validation

Given the limited size of the dataset, k-fold cross-validation is used in the experiments in order to assess the capability of the proposed architecture to extrapolate to unseen sequences. The process consists in splitting the data in k groups and then performing k iterations of training, each one leaving a different group for the validation and test sets. That way, every piece of data is used in training $k-2$ times and used in validation and test 1 time. For every iteration, the chosen performance metrics are recorded and later on summarised in order to show a general view on how the model will learn from the data. This process is commonly used to evaluate machine learning models over a representative subset of the dataset. In this case, it is a good measure of the model's performance due to the limited size of the dataset available.

Since different frames throughout the same video have similar features, the splitting of the data was done at the sequence level in order to ensure that the training, validation and test sets contain disjoint data. Therefore, 5-fold cross validation was applied to the dataset so that at each iteration, the training set is composed of 3 videos and the validation and test sets contain 1 video each, as defined in Table 3.5.

Split \ Video	DSC_1089	DSC_1098	DSC_1104	DSC_1107	DSC_1109
1	train	train	train	validation	test
2	test	train	train	train	validation
3	validation	test	train	train	train
4	train	validation	test	train	train
5	train	train	validation	test	train

Table 3.5: 5-fold cross-validation structure

Each experiment will be performed over every split and then the mean of the metrics will be calculated in order to compare it with other experiments. It could be the case that a specific experiment works best in one concrete fold, therefore calculating the mean of all the folds returns more generalistic metrics.

3.3.4 Initial parameters and experimentation procedure

Due to time constraints, not every permutation of the parameters of the network can be experimented with. Therefore, some of this variables were fixed from the beginning and the approach taken with regards to the rest of the parameters was incremental. Firstly, the parameters considered to need a more thorough experimentation were isolated and ordered by their expected importance. Each parameter was tested and it was fixed to the best value found for subsequent experiments until every one in the list was tested.

The parameters considered less crucial or more straightforward were fixed at the beginning or experimented with shortly, not enough to constitute a section in this document. Some of these values are:

- **Batch size:** The model can process several images at a time, the amount of images to be processed at once is called batch size. Since the network computes the loss and back-propagates the gradient for each batch (and not each image), batch size was fixed to 1 in order to maximise the amount of updates of the network per epoch. This parameter is also determined by the memory of the machine, as a complete batch of images need to be loaded into memory to be processed. In the used computer the maximum value possible was 5. Testing was done with batch size 1, 2 and 5, but the best results were achieved for batch size 1. To process one image at a time but only do back-propagation once per video (every 10 images) was also tested, but it did not yield better results than straight batch size 1.
- **Number of epochs:** An epoch is constituted by passing the entire training set through the network. Experiments were carried out with 250, 400 and 1000 epochs. It was found that the network starts to converge at around epoch 100 and the system is mostly stable at around epoch 140, therefore the total number of epochs set for the experiments was 150.
- **Image size:** The network takes as input square images, the size of the square can be altered and it will affect the maximum FPS at which the model can function. Even though this parameter can be changed, it is not as straightforward as one may think, since the image size is directly related to the internal structure of the network, and not every image size is eligible. Therefore this parameter was left as its default value of 416px.

The following sections proceed to explain the basis for experimentation over the rest of the considered parameters. The experiments will be performed in order, explaining the premise of the experiment, the values tested, as well as the analysis of the metrics and the best value found. Every performance graph that is shown in this chapter corresponds to the mean of every cross-validation fold with that concrete experiment's parameters. The most appropriate model for each experiment will be chosen by considering the validation performance metrics, since they represent the ability of the model to process never-seen-before data.

3.3.5 Pretraining

The first experiment that was conducted corresponds to the initialisation of the model with a set of pretrained weights in the convolutional layers of the network. Both Linder-Norén [19] and Redmon [20] recommend the usage of pretrained weights from a darknet53 model trained on the ImageNet dataset. It was considered that, due to the difference between ImageNet and the custom HET-CAM dataset developed in section 3.2, maybe the pretrained weights do not play an important role in the training process of the model required for the problem at hand.

The ImageNet dataset consist in images of all sorts of common things: teapots, phones, zebras, cars...It does not contain any information relevant to the HET-CAM problem and, since this domain is very specific, testing needed to be done in order to verify what is better for the training process. The alternative to using the pretrained weights is initialising every weight to 0. The network underwent training with both of these configurations and the results obtained are stated in figure 3.7 (page 32). Both experiments (pretrained and non pretrained weights) were performed over the same set of parameters, the only one that was changed was the initial weights of the network.

The graph shows curves with approximately the same shape regarding the training metrics, which means that the pretraining does not play a part in the rate at which the model evolves. However the curves defined by the pretrained model have better values in every step of the process, which implies that an appropriate model will be reached at an earlier epoch. This suggests that initialising the weights with those of the pretrained model, even though it was trained over a different dataset, helps with basic object recognition.

To analyse the performance of the model to never-seen-before images, the validation set is passed through the model at the end of each epoch and the metrics computed are shown in figure 3.8 (page 33). Better values of precision and recall are displayed for the pretrained network, reassuring the results given by the training metrics. Not only that, but the pretrained network yields a much more stable number of false positives in the negative test, thus confirming the idea that pretrained weights do contribute positively to the model's performance.

By comparing training and validation metrics, it can be seen that, even though the model keeps evolving trough the whole process, the validation metrics stabilise at around epoch 80.

By letting the network train more, overfitting occurs, that is, the system learns to detect the concrete haemorrhages in the training set, thus not being suitable for any other videos. That is why a stagnation of the validation metrics is observed and a slow decrease over time is expected. The model is less overfit when both the training and validation metrics are better.

At this point several lessons were learned. Firstly, by the recall peaks observed in both pretrained and non-pretrained experiments, it can be seen that the pretrained model not only achieves its peak performance at an earlier epoch but it also has higher performance overall. And secondlyly, the inclusion of pretrained weights does not affect the rate at which the system evolves, that is, the shape of the curves is the same in each run, other than the pretrained run produces superior results. Thus, it can be concluded that initialising the model's convolutional layers with pretrained weights from a darknet53 model trained over the ImageNet dataset is advantageous. Therefore, all subsequent experimentation will be performed over a system that uses such configuration.

3.3.6 Learning rate

The training process of a deep learning network consists in modifying the weights of the layers in order to minimise the loss, to accomplish this the gradient descent algorithm is used. The learning rate is a parameter that impacts the amount of variation that the weights can attain in each step, and choosing an appropriate value is of the utmost importance for the correct development of the training. Having it too low may cause the network to converge to a suboptimal local minimum and having it too high can cause the training to be unstable, never converging to the desired solution.

From the last experiment, it was discovered that the network overfits quite rapidly, therefore several experiments were conducted by changing the value of the learning rate to check if the network can achieve better validation metrics before overfitting. Since the program uses the Adam[21] algorithm to alter the weight's values, what is usually called learning rate in other optimisation algorithms in this case is called alpha (or effective step size), but it represents mostly the same concept as explained in the previous paragraph. Using Adam means that the learning rate will not be constant in every weight along the whole training process, instead it starts at a fixed value (alpha) and then it changes and adapts to the gradient of that specific weight, achieving better performance than other stochastic gradient descent optimisation algorithms. In the context of these experiments, the default value for alpha is 10^{-3}, as suggested by the paper[21], but in order to find if this value is the best fit for the problem at hand, experiments were run with higher and lower powers of 10. The results of the experiments can be observed in the following graphs (Figure 3.9, page 34 and Figure 3.10, page 35).

It seems that altering the learning rate of the model does not change its behaviour at all.

This may be caused by the pretrained weights. The model is initialised with values of a trained network which may cause the gradients to be small from the beginning, this is not expected to happen with models that start from a blank slate. If the amount of variation that needs to be introduced in each step is smaller than the smallest learning rate, all experiments would yield the same results.

Since it was discovered that altering the learning rate does not matter, the default value recommended by the Adam algorithm[21] of 10^{-3} will be used.

3.3.7 Object detection factors

The loss formula that is used by the network [13] is comprised of several factors, two of which weight the importance of detecting every object (even though it might be classified as a different class) and not making a detection where there is not an object. From now on these factors will be called object loss and non-object loss. Both of them have associated a fixed constant in order to balance the priority between finding all the target objects and not associating an object detection to the background of the image. These constant values will be called object scale and non-object scale respectively.

Since typically there is a lot more background area that there is object area in an image, it is clear that non-object loss will be a lower value most of the time, because a greater part of the detected background will coincide with the expected background. By default, object scale and non-object scale are weighted 1 to 100 in order to make non-object scale have a bigger impact in the total loss value. A reduction on this proportion would mean that the model would prioritise finding all the objects, taking less care in marking parts of the background as objects.

Due to the marking procedure to annotate the dataset, most images count with haemorrhage spots that were not tagged and so, even though they are considered a part of the background in the dataset, they are actually wanted objects to be detected by the final system. This means that, as the dataset only counts with the minimum set of haemorrhages that want to be detected, recall must be as high as possible. On the flip side, as the dataset lacks exhaustiveness when annotating the images, there are haemorrhages in the pictures that are not annotated, therefore the precision metrics recorded will be lower than the actual precision of the system. In order to prioritise recall over precision different tests altering the proportion between object and non-object scale were performed. If the proportion is weighted towards the object scale side, the recall will be prioritised, while, if the ratio is more on the non-object side, precision will increase. Due to the nature of object and non-object loss, it is considered that a 1-100 ratio is balanced, so tests will be performed leaning towards the object's side. Different ratios were tested and the results are printed in figures 3.11 (page 36) and 3.12 (page 37).

As it was expected, lowering the non-object scale decreases the penalty for getting false detections, the false positive rate increases and thus the precision is usually lower. This behaviour happens because, if the model returns a big number of random detections, just by chance some of them will overlap with the targets and be considered as true positives thus achieving the higher values of recall as observed in the 1-25 ratio experiment. Not penalising all those extra detections would lead to a lower precision. Nonetheless, in the context of the dataset being studied, inferior values of precision do not necessarily mean bad performance, since, as it was said before, not all haemorrhages are annotated. In fact, in figure 3.13 (page 38), it can be observed that a good number of false negatives do correlate to haemorrhage spots that are not annotated in the dataset. This concrete example only counts with a few "true" false positives, for example the one in the lower left part of picture that just captures the top side of a healthy blood vessel and a flat background.

The difference is more clearly observed in the training metrics, where the different ratios are ordered in the recall metric, and reverse ordered in the precision metric. Nonetheless, in validation the difference is a lot more blurry. It is obvious that this parameter does not take a part in the number of false positives in negative images (just an anomaly is observed in 1-25 ratio epoch 30). But the decrease in precision observed in training is not clearly visible in validation. Looking at the validation metrics, it was decided that the 1-25 ratio is the most appropriate, as it achieves the highest recall and F1 score in validation but it does not see a high decrease in precision. Note that the 1-1 ratio was tested, but the network returned such a high amount of predictions that training was taking too long and the experiment was aborted.

To conclude, it was decided to keep the object and non-object scale correlation at a 1-25 ratio as a way of boosting the recall of the final system without hurting the precision and processing time too much.

3.3.8 Data augmentation

Data augmentation is the process of randomly changing the images in the dataset, increasing the number of training samples in order to improve the generalisation of the network. This process is specially useful when the amount of available data is limited, as in the HET-CAM problem. Some techniques involve mirroring the image in the horizontal or vertical axis (or both), scaling, rotating, or changing the colour of an image. In the context of this problem, due to the limited amount of training data, overfitting occurs quite rapidly so different augmentation methods were tested in order for the network not to learn the specific images of the training set.

First of all, several geometric transformations were applied. In order to pass the images to the system they are padded to conform a square and then scaled to the desired size. This

size starts at its default value of $416px$ and is changed every 10 batches in random multiples of $32px$ up to $\pm 96px$. These $32px$ intervals were chosen because they are compatible with the internal structure of the network, specifically the downsampling or pooling layers, where the input is reduced to a fraction of its size. If the new size after downsampling or pooling is not an integer the model can have troubles or even crash. Apart from that, each image in a batch has a 50% chance of being flipped horizontally.

The video acquisition processes and the nature of the captured data cause luminosity and contrast heterogeneity within the HET-CAM frames. In order to have a more extensive representation of the data variability, changes in the colour of the image were also considered. In order to separate the raw colour from the lighting component, the images were converted to the HSV colour space, where they are split in the components of Hue (colour), Saturation (intensity) and Value (lighting). A script was developed to study the predominant H, S and V values in different images. The program orders the H, S and V values by number of appearances in each picture and then truncates that list to get the top values that account for $>= 50\%$ of the image. It was concluded that the predominant set of values across all of the videos of the dataset is composed of: Hue values ranging from 8º..24º, Saturation from 0.768..0.941, and Value from 0.549..1. The result of applying a random colour transformation with this values to a frame can be seen in figure 3.14 (page 38)

Therefore in order to alter the pictures in a way that is not incoherent with the values observed in the dataset, fixed amount of variation in the HSV values was introduced. A random variation of ± 8 H, ± 0.1 S, and ± 0.1 V was applied to the images with a 50% chance of happening. Having these random HSV transformation can help to artificially produce images that represent different lighting conditions, different egg colours, etc. The final state of the data augmentation pattern is the following:

- Every 10 batches the size of the image is changed from its initial size to up to $\pm 96px$ in intervals of $32px$
- A quarter of the time the image is left as is
- A quarter of the time the image is flipped horizontally
- A quarter of the time the image is changed colour in the HSV space: $\pm 8H$, $\pm 0.1S$, $\pm 0.1V$
- A quarter of the time the image is both flipped and changed colour

Experimentation was performed to compare using all of this augmentations with not using any of them, the results can be seen in figure 3.15 (page 39). Contrary to the expected results, this amount of data augmentation does not affect in any way to the training process.

Not only that, but, as it can be observed in figure 3.16 (page 40), a very slight decrease in performance is caused by the use of data augmentation. Thus it can be concluded that data augmentation does not take a part on the performance of the network under the current circumstances. It is conjectured that data augmentation does not make a big difference due to the very limited size of the dataset, since there are just 10 validation images. It is expected that with bigger datasets, data augmentation will help the model abstract the characteristics of the images and therefore have better performance. Nonetheless, data augmentation is usually recommended [13] and therefore, as it does not help nor harm the performance of the model, it will be used in the final configuration.

3.3.9 Final network configuration

Throughout this section the impact of changing several parameters in the configuration of the network was studied. To conclude, a summary of the chosen configuration and the decisions made is presented.

First of all, the chosen implementation [19] advised to use pretrained weights for the convolutional layers of the network. Since the algorithm is commonly used for general object detection, specifically having different classes of objects to be recognised, it was considered that loading the pretrained weights might not play an important part in the context at hand, where only one type of class is present (a haemorrhage spot). From the experimentation carried out, it was concluded that loading the pretrained weights is advantageous, even if the dataset used has only one class and is very different to the ImageNet dataset.

From the pretraining experiment it was noted that the model overfits in a rapid manner and the validation metrics reach their maximum values in a low number of epochs. In order to give the model more time to evolve before overfitting, expecting an increase performance, lower learning rates were tested. It was found the default value of 10^-3 is appropriate for the problem at hand, since, lowering it did not cause any change in performance.

Since the dataset is not exhaustively annotated, the measurement of false positives and precision rate needed to be carefully studied. It turned out that the network did detect some haemorrhages not tagged in the dataset and thus categorised them as false positives. The importance of finding all the annotated objects with respect to not finding too many false positives is balanced via the object and non-object scale factors in the loss formula. Experimentation was carried out varying these values in order to get the recall-precision balance appropriate for the context of this problem. The results pointed to a 1-25 ratio between object and non-object scale, to prioritise recall but still do not get a very low precision and number of ”true” false positives.

The original YOLO paper [13] states that data augmentation was used in the training process, so the advantages of using it or not using it were studied via experimentation. The

videos were transformed to the HSV colour space and their colour palettes were studied in order to set the bounds for random variations to be applied to the images in the dataset. Apart from that, several geometric transformations such as scaling and mirroring were applied to the images. The results were uncertain, data augmentation did not play a role, neither positive nor negative, in the performance of the network. This may be caused by the limited size of the dataset, so the final model configuration counts with data augmentation expecting that it will grant a positive outcome if the size of the dataset increases.

As a final note, the model does not achieve the standard values of recall, precision or f1 score that are expected for a state-of-the-art artificial intelligence system. It is, however, capable of distinguishing between positive and negative videos, as it can be seen from the number of false positives in negative test performance metrics. Taking into account that the validation metrics are calculated over several frames of different videos, but the program being developed studies the correlation between consecutive frames, the most important feature is that the model reacts similarly in similar frames in order to build a well formed graph of the evolution of the irritation process.

Therefore, with the final configuration of the network, the system's performance needs to be checked by analysing its response to the test set. Since this set was not used to guide in any way the selection of the parameters, the metrics given by analysing it represent the real world performance of the system. Same as with the rest of the experiments, all the cross-validation folds were trained. Then, from each fold, the best epoch was chosen and passed through the test set in order to measure its performance.

As said before, a higher recall is a priority while choosing the best model in each fold. But, due to the high variability noticed in this metric along the experimentation, it is consider that basing this decision solely on the recall would be very volatile. So, in order retrieve a network with high recall but make sure that this value is not an anomaly, the sum of recall and f1 score was computed for each epoch and the highest value in each fold was selected. As the f1 score is biased by the precision, taking it into account ensures that the chosen model does not achieve a high recall value just by returning a big number of detections.

With the epochs selected for each fold, the metrics over the test set of that fold were computed. Table 3.6 (page 30) shows the chosen epoch as well as its performance in the test videos, both positive and negative.

Note that, even though measures were taken to try and avoid picking an anomaly as best epoch of a fold, the result returned by fold 1 is indeed an anomaly. This reflects what was explained in the object scale experiments where, if the loss does not penalise the appearance of large amounts of false positive detections, great recalls can be achieved just by returning a large amount of random bounding boxes. In fact, this anomaly causes the average amount of false positives in negative tests to increase to 22 whereas, if this fold ignored, it would be just

Fold	Best Epoch	Recall	Prec.	F1	neg. test FP
1	70	0.837	0.157	0.264	92
2	90	0.831	0.414	0.553	4
3	40	0.360	0.487	0.414	9
4	60	0.683	0.336	0.450	2
5	50	0.480	0.330	0.391	3
	Avg:	0.638	0.344	0.414	22

Table 3.6: Final configuration's best epoch test metrics

4.5. In figure 3.17 (page 41), which corresponds to the test performed by fold 1, this concept is clearly observed. It is clear that this kind of behaviour is of no use to the HET-CAM problem, but a positive note to take from this test is that it only happened in one of the folds, which means that it is not the working procedure of a standard trained network.

Considering the amount of haemorrhages tagged in the dataset and the metrics achieved,even taking into account fold 1, it is clear that the model will be capable of distinguishing very clearly between positive and negative videos. Take, for example, fold number 3 which only has 0.487 recall. That means that out of the 211 haemorrhages marked in video DSC_1098, the model correctly predicted around 102 of them, compared with just 9 false positives detected in the negative test. This means that even though the metrics values are low, in this context they are more than enough to determine if a video is positive or negative.

Not only that, but the model also returns a representative amount of the haemorrhages in each frame, which can be used to help in determining the toxicity of the drug. In this context, it is important that the system makes similar predictions in similar frames, as the HET-CAM study relates to the evolution of the irritation process, not the shear number of bleeding area. Therefore, low performance metrics do not imply that the model is unusable.

The following images correspond to the detections on the test set of fold 4, where the false negatives are highlighted in blue, false positives in red and true positives in green. In figure 3.18 (page 41), it can be observed that the network does not detect all the haemorrhages targeted, but a good number of false positives do correspond to bleeding areas, double detections our do not achieve the 0.5 IoU to be considered true positives. This kinds of values hurt a lot the numeric performance of the network, but upon visual inspection of the images, it can be seen that the amount of "true" false positives does not correspond to a 0.45 precision. These images also show that the proportion between the confusion matrix values is mostly conserved as the video passes, therefore, from an bleeding evolution perspective, the model is expected to return a similar evolution curve when studying drugs that cause similar processes in the membrane.

As a final note, emphasise once again that, in the frames shown in this section, a good number of detections considered false positives are in fact true positives. Thus, making the

true recall, precision and f1 scores better than the values displayed. Nontheless, this low theoretical values do not harm the performance of the network when analysing the evolution of bleeding areas.

3.4 Bleeding evolution module

The second module of the proposed methodology consists in gathering the information provided by the bleeding detection module throughout all the frames composing the HET-CAM videos in order to extract objective information regarding the bleeding evolution. It was decided that the final program will be delivered as a Python script *analyse.py* that allows the user to pick a video, analyses it using a trained model, and then shows number of detections and amount of bleeding area graphs in a Matplotlib[22] figure. By providing a script instead of an enclosed executable, the customisation of the program and output is maximised, allowing the users to change it to their needs. It is also recommended to have Pytorch installed with CUDA, as it hardware accelerates the analysis process and makes it a lot faster. This script does not count with a dedicated section in chapter 4 because it is a composition of parts of other scripts.

In order to feed the video frame by frame to the network, the (Pytorch) Dataset class was extended. The VideoDataset developed uses OpenCV[23] to read and return each individual frame. This can be easily changed to read a live video feed. Reading the whole video frame by frame can be slow, therefore, a parameter was introduced to ignore regular intervals of frames between samples.

The program first loads the trained model into memory and then opens the video selected by the user as a VideoDataset. It iterates over the dataset, analysing each individual frame and saving the number of detections and the area of bleeding. Since bounding boxes tend to overlap in areas with a high density of detections, the procedure used to calculate the total haemorrhage area is the union (instead of the sum) of all the areas detected. By doing this, counting twice the overlapping of areas is avoided, and a more reliable metric is calculated.

Finally, a Matplotlib figure is show with two subplots corresponding to the bleeding area and the amount of haemorrhages detected in each frame of the video with respect to the time in seconds(Figure 3.19, page 42). This allows the user to graphically see the evolution of the irritation process and also be able to determine if a video is positive or negative. Both graphs are bound in the X axis, which makes easier the exploration of both of them at the same time.

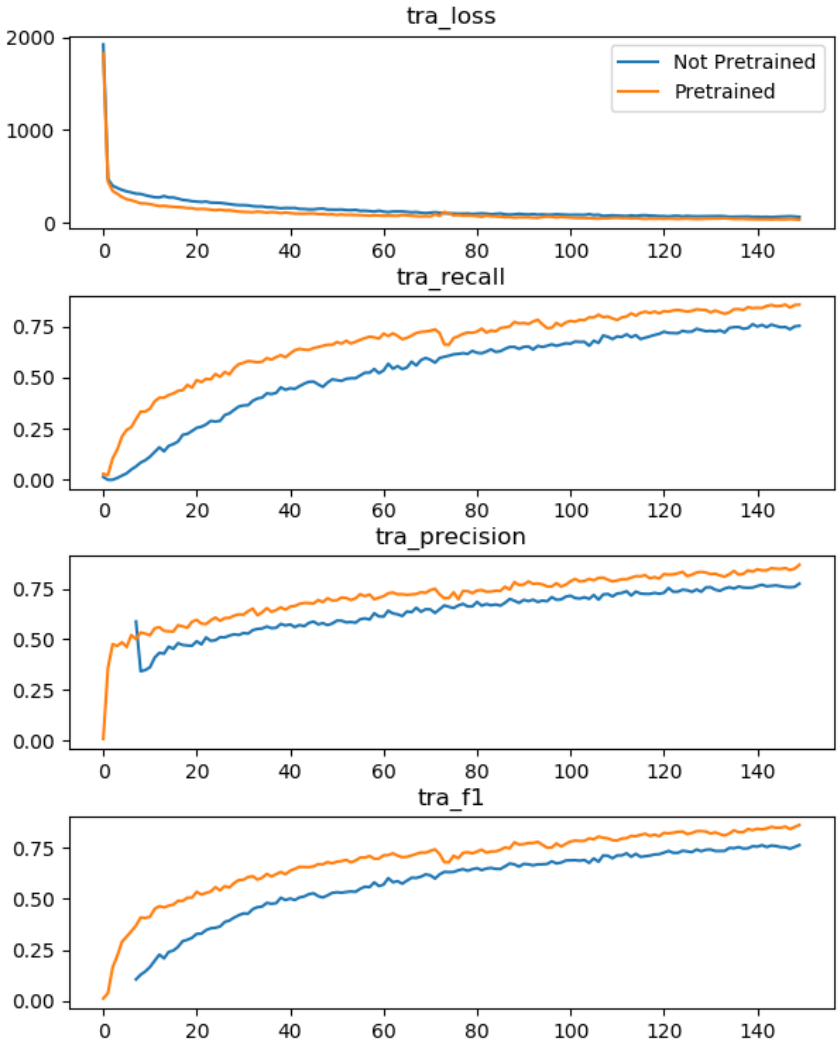

Figure 3.7: Pretraining experiments' training metrics

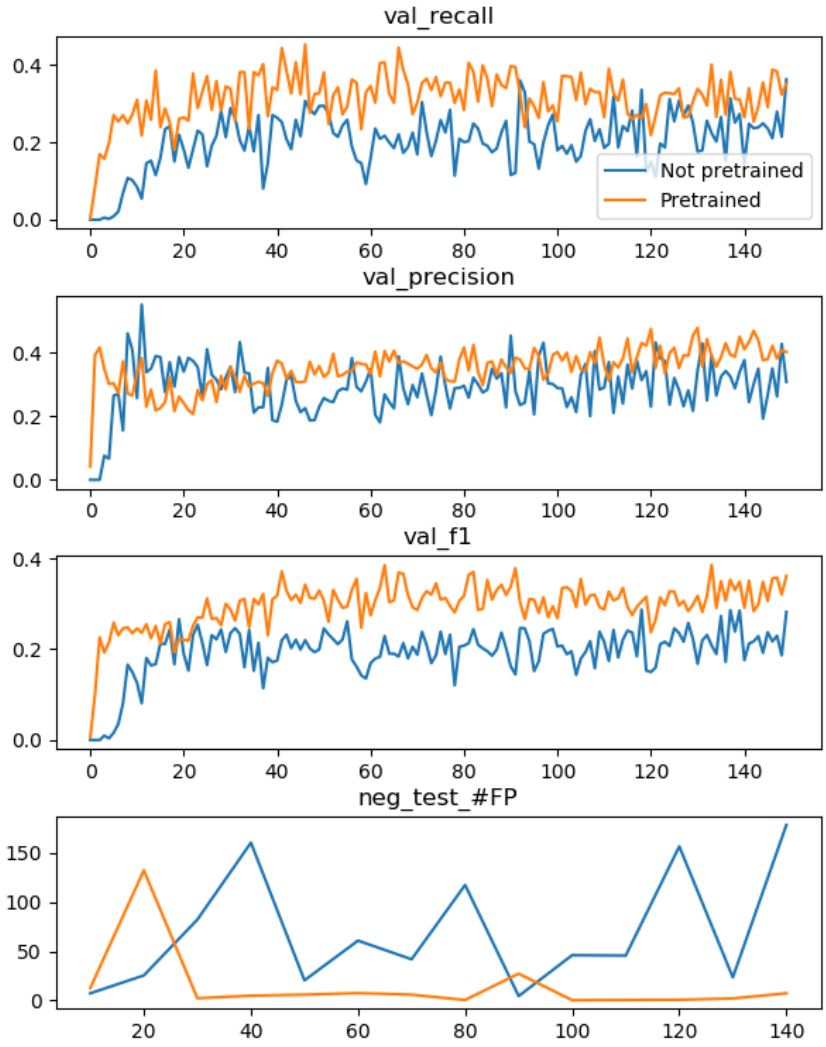

Figure 3.8: Pretraining experiments' validation metrics

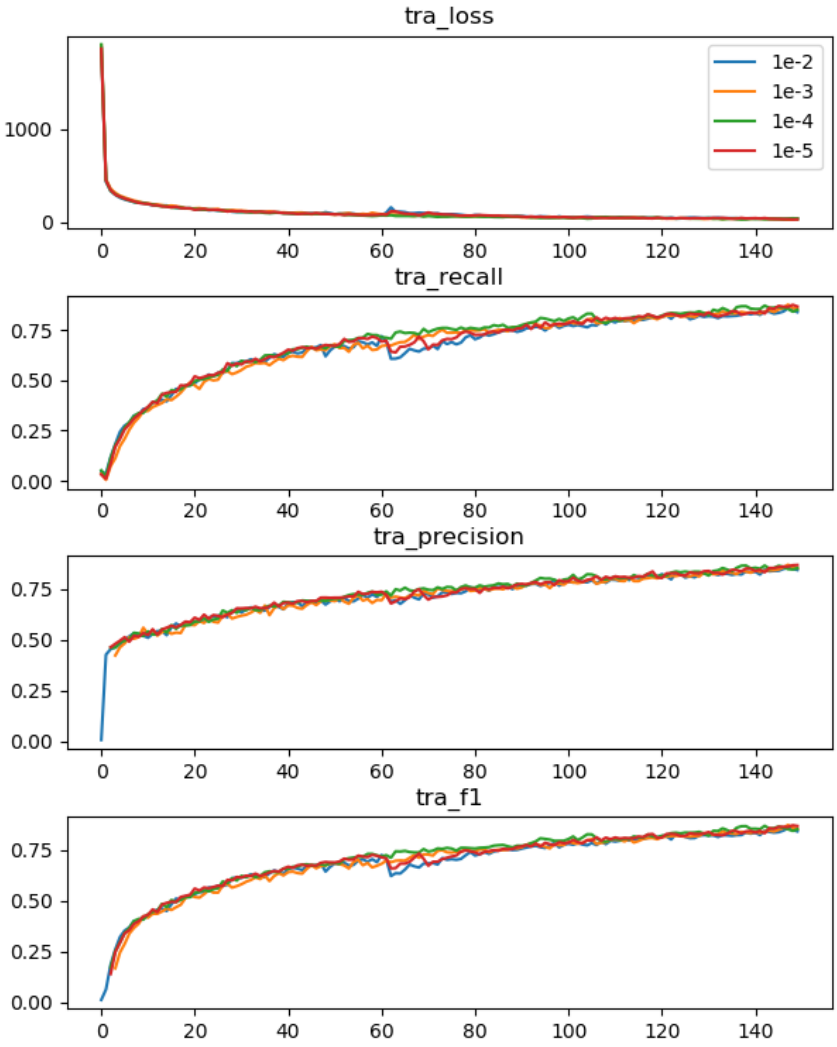

Figure 3.9: Learning rate experiments' training metrics

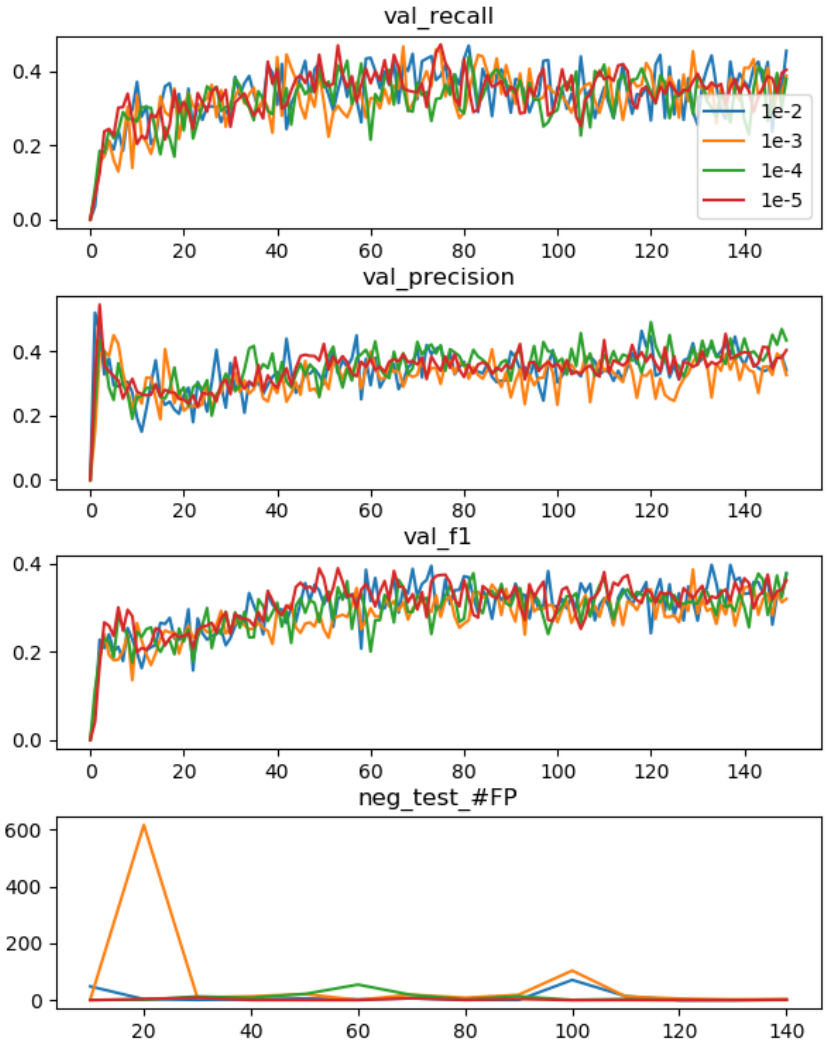

Figure 3.10: Learning rate experiments' validation metrics

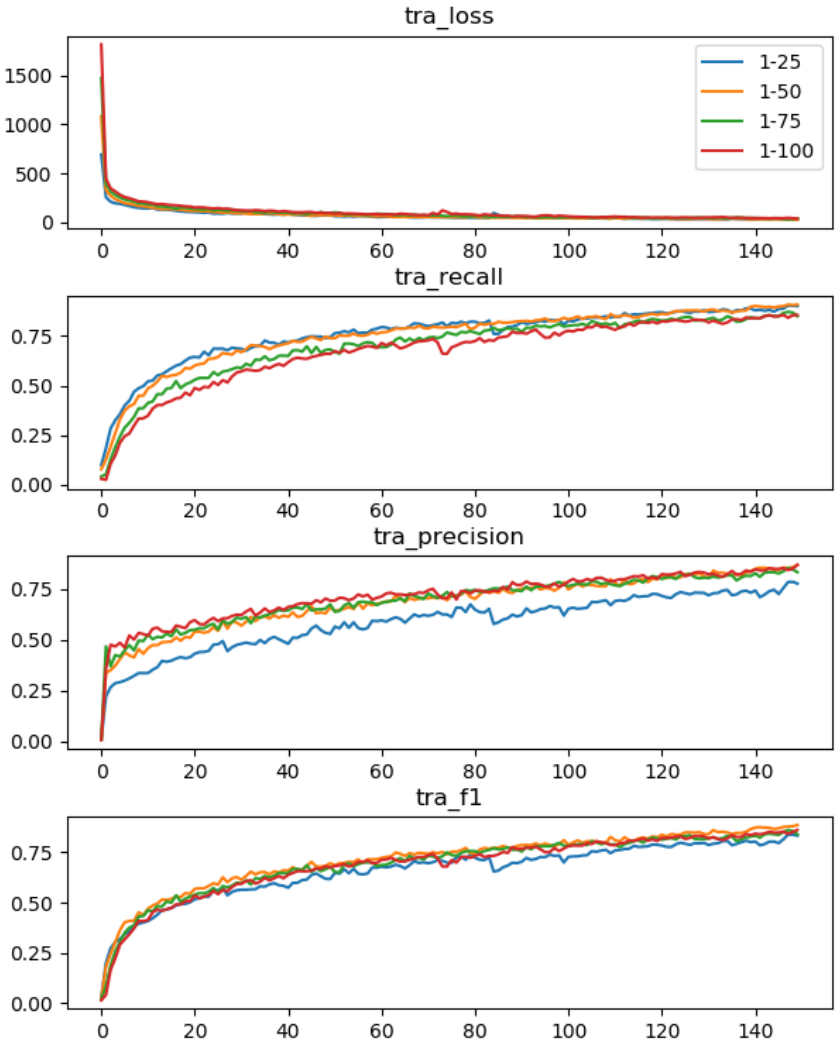

Figure 3.11: Object and Non-object scale experiments' training metrics

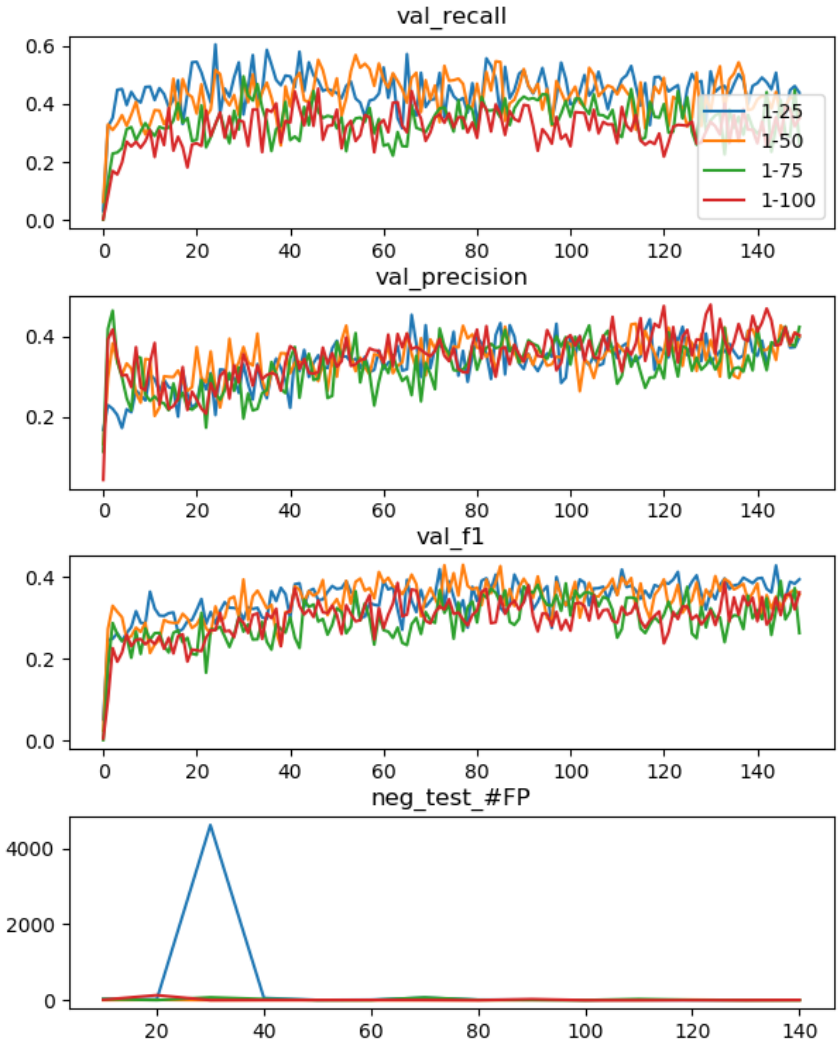

Figure 3.12: Object and Non-object scale experiments' validation metrics

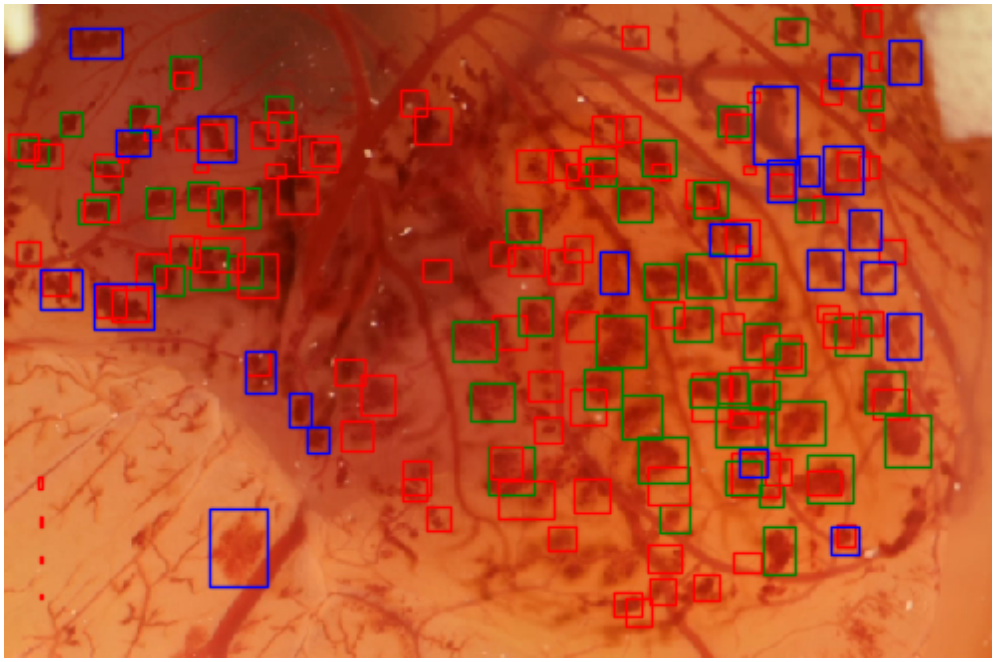

Figure 3.13: Example of model detecting more haemorrhages than the annotated by the dataset

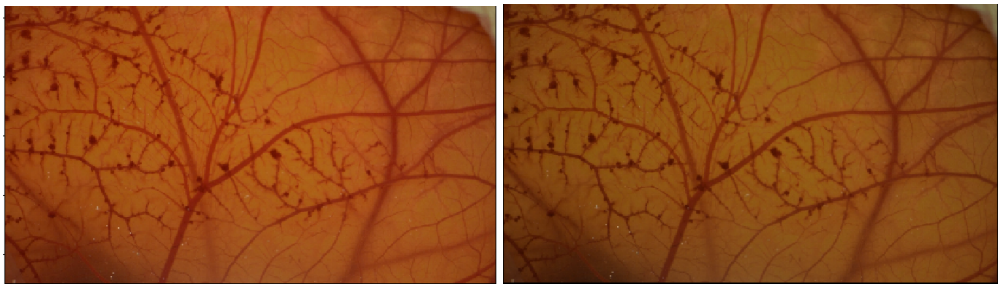

Figure 3.14: Example of a random transformation in the HSV colour space following the stated parameters

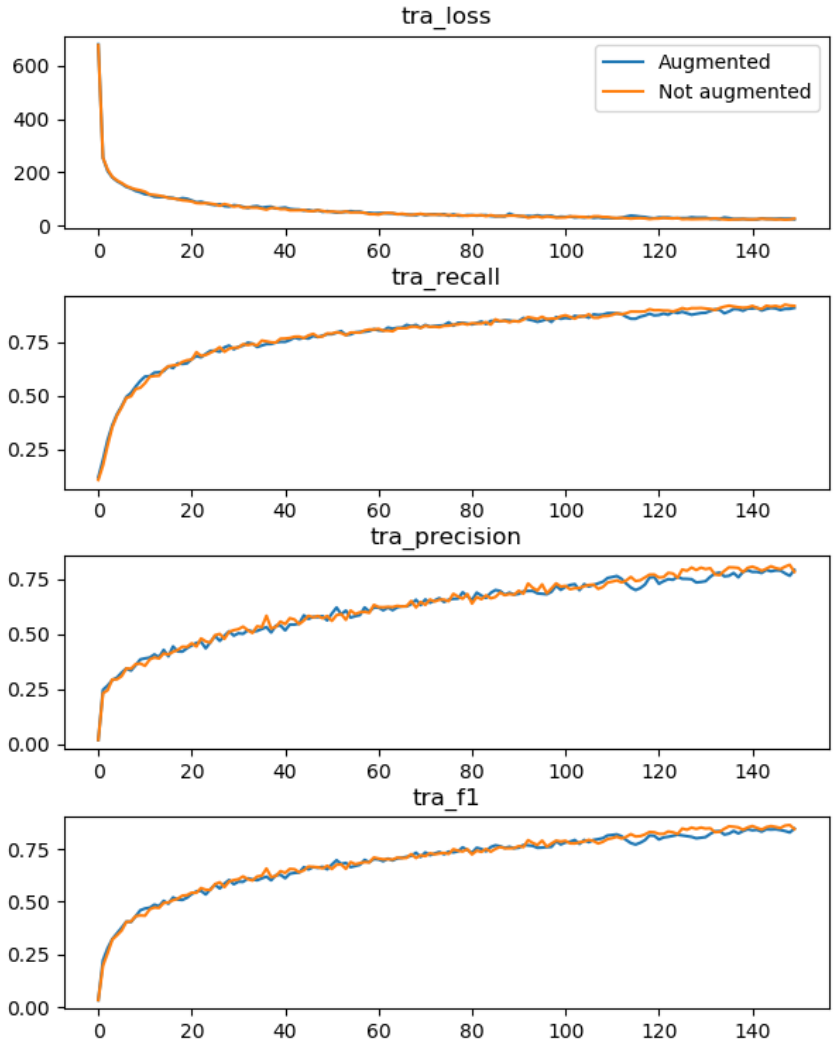

Figure 3.15: Data augmentation experiments' training metrics

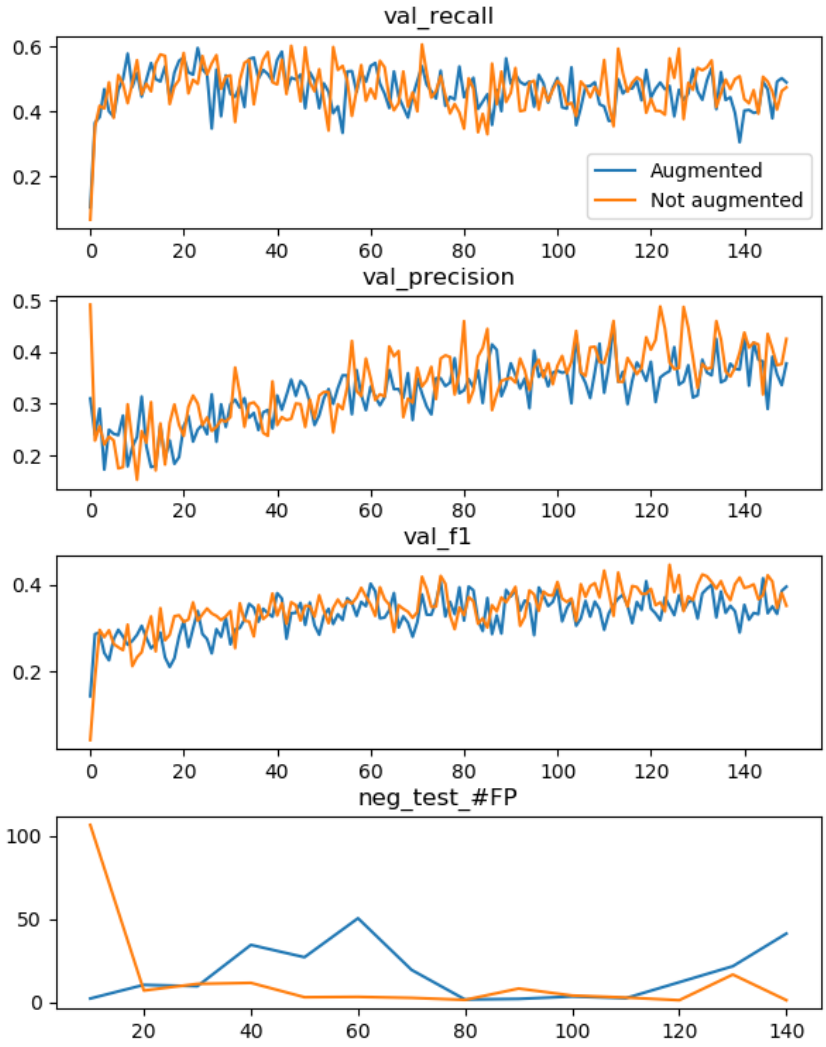

Figure 3.16: Data augmentation experiments' validation metrics

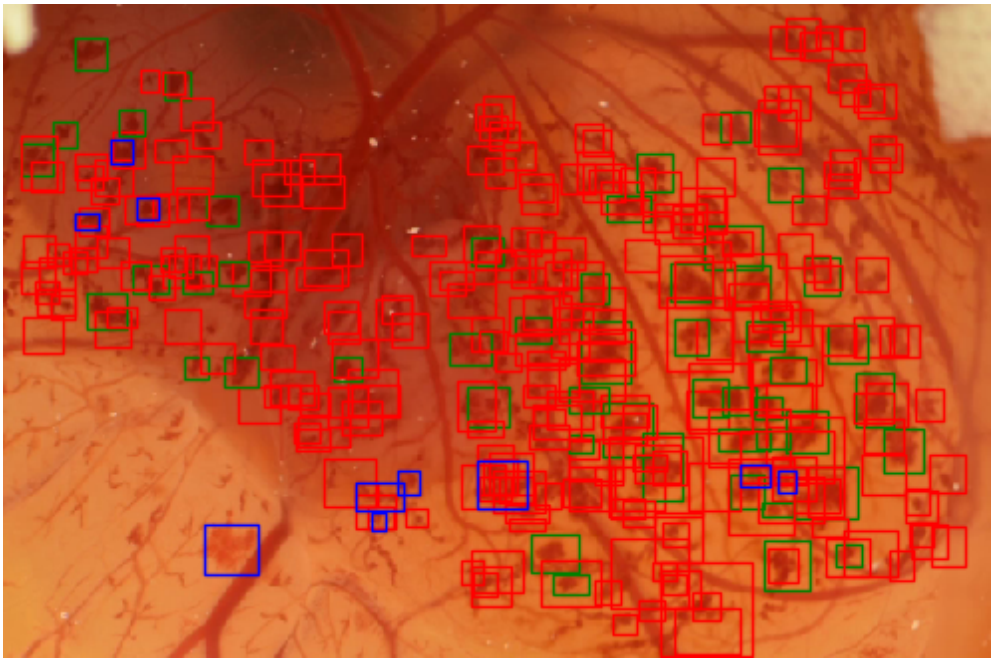

Figure 3.17: Fold 1 test results of a frame: TP(green), FP(red), FN(blue)

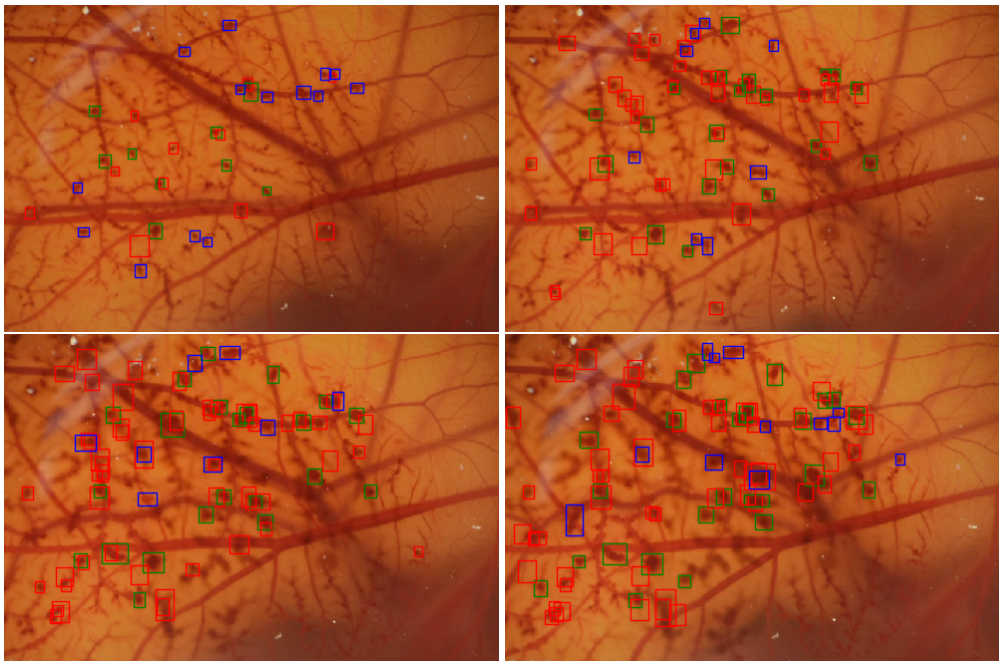

Figure 3.18: Evolution of bleeding in different frames: TP(green), FP(red), FN(blue)

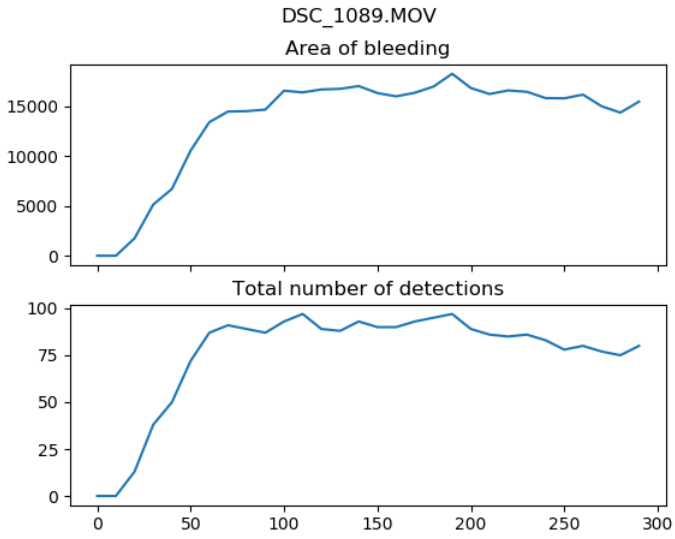

Figure 3.19: Output of the video analyser script *analyse.py*

Chapter 4

Software Development

For the accomplishment of the project's objectives, several scripts and programs need to be developed. This chapter will detail the characteristics of each of them as well as the development process of the most complex ones from requirements, to design and finally their implementation.

The Python language was chosen for all of this operations because of the script-like nature of the problem. Its ease of use and the extensive amount of libraries like OpenCV [23] or Matplotlib [22], that make image processing and navigation easy and straightforward.

4.1 Image extraction and pre-processing

To perform the process of building an annotated dataset from the source videos, a number of steps need to be followed. First, the frames of interest need to be extracted from the videos and then each individual frame's areas of bleeding need to be annotated following the specification described in section 3.2.2.

The source of data that is available consists on a series of *.mov* video files, whereas the input data of the AI system are image files. Therefore, a program needs to be developed that transforms the videos into suitable images to be used by the system. At first, the images extracted from the videos were saved and used by the network, but during the experimentation process, it was found that the aspect ratio of the images may take a part in the correct behaviour of the model. Since there are videos in two different aspect ratios and the chosen YOLO implementation takes square images as input, the widescreen 16:9 frames need to be trimmed in order to normalise the aspect ratio of all the images of the dataset to 4:3. This operation allows to preserve the original aspect of the frames, preventing the distortion derived from resizing the samples to pass them to the system. Since this was found after the video frames were extracted and filtered, the two different scripts detailed ahead were not developed as just one program.

Description	A Python script that converts a video to a series of *.png* images, skipping a certain number of frames between samples
Input	- Path to the video that wants to be converted - Number frames to be skipped
Output	A folder with the extracted images in the same directory as the input video
Comments	

Table 4.1: Specification of *FrameExtractor.py*

Description	A Python script that converts a set of 16:9 images into 4:3 aspect ratio. It saves the right and left sub-images for each input image
Input	- Path to a folder of images that want to be converted
Output	A folder within the selected folder containing the new 4:3 images
Comments	

Table 4.2: Specification of *AspectRatioConverter.py*

4.1.1 Design

First of all, the script that extracts frames from the videos, *FrameExtractor.py*, was designed. In order not to take too long, the amount of frames that are extracted per video is controlled by a parameter, allowing the user to skip a certain number of frames between saved images. The specification of the script is detailed in table 4.1

At a later time, the need to change the image's aspect ratio arose. It was decided not to redesign the previous script and re-process all the videos because the frames of interest were already filtered. Instead, a separate script, *AspectRatioConverter.py*, was designed to transform a set of 16:9 images into 4:3 aspect ratio. The images' 4:3 resolution components will be calculated by keeping the current height constant and calculating the new width to get a 4:3 resolution. With that, two sub-images will be saved per input image, corresponding to taking the new width from the left and the right borders of the orinigal image. The specification of the script is detailed in table 4.2

With this process, the centre portion of the original image appears on both right and left sub-images, as it is seen in figure 4.1 (page 45).

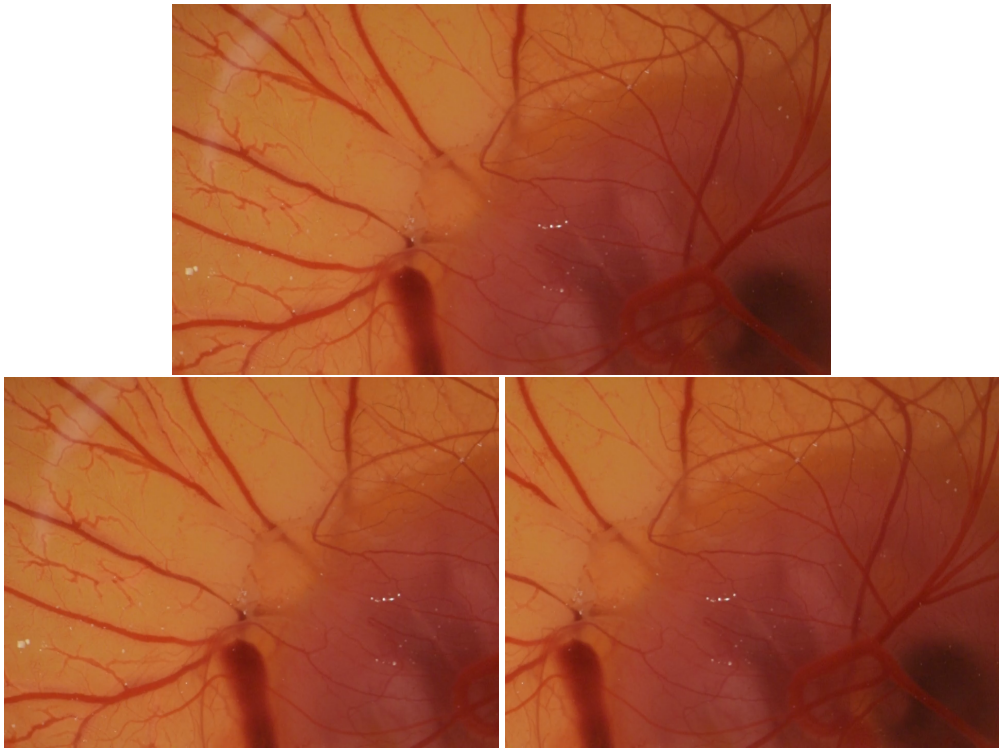

Figure 4.1: Aspect ratio converter: Input image - Left sub-image - Right sub-image

4.1.2 Implementation

The implementation of these two scripts was very similar, using TKinter to graphically ask the user for the parameters, be the video or the folder of images to be converted. Then, the whole behaviour of the script was encapsulated in a function with the specified inputs and outputs. The library chosen to open and save the videos and images was OpenCV[23]. Since both of the scripts are very simple and do not count with any custom class or other complicated structures, no use case, class or other kinds of diagrams were drawn.

In case of *FrameExtractor.py*, first the user selects the video from which to extract the frames, then the image processing library OpenCV[23] is used to read the video and save the frames as *.png* images. To make the script more user friendly, the choosing of the video is performed over a FileDialog from the library TKinter. The length of the interval that separates two consecutive samples is a configurable parameter of the program. The script saves the images in a folder named *[video_name]-frames* in the same directory as the chosen video and each image has the name *[video_name]-frameX* where X is the frame's index in the video file.

Same as with the last script, 4.2 uses OpenCV for image processing and TKinter to allow the user to choose a folder via a GUI. The program calculates the new dimensions of the images and saves them individually under the name *[img_name]-[L|R]* in a folder called *converted*

within the selected directory.

4.2 Image marker

The program described ahead was developed to act as a visual interface to help identify the bleeding areas and annotate them as described in section 3.2.2. Its behaviour is best described by the use-case diagram depicted in figure 4.2 (page 46)

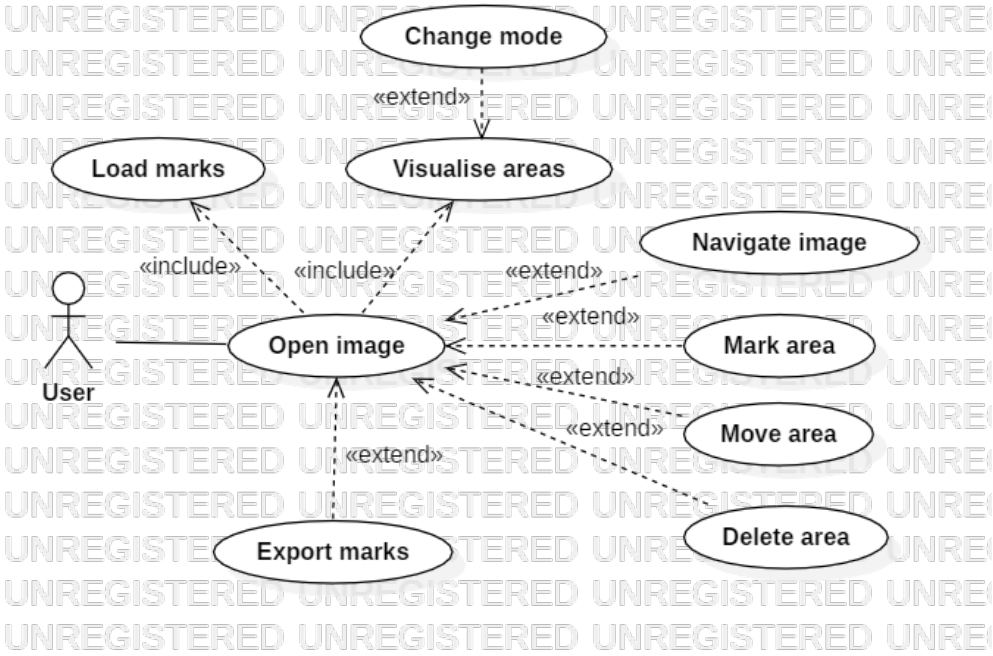

Figure 4.2: Image Marker use case diagram

- **Open image:** The program opens a file system explorer window to help the user navigate to the image that needs to be annotated.
- **Load marks:** The system tries to open the text file that describes the areas marked in the picture; if it does not exist, the program continues.
- **Visualise areas:** The user is able to see all the objects marked in the current image.
- **Change mode:** The visualisation mode can change between showing the centre point of each area, or showing a rectangular outline of the bounding box.
- **Navigate image:** The user can move and zoom to focus on different parts of the image to help with the marking process.

- **Mark area:** This can be accomplished in two different ways. The first one consists of a fixed size area centred around the point where the user clicks the image. The second one is a two-click process where the user needs to mark the top-left corner first and then the bottom-right corner of the area that surrounds the object.
- **Move area:** The user can click and drag a marked area to move the mark around, preserving its size. This functionality is implemented to help the user fine-tune the location of a mark in case the area is not perfectly centred initially.
- **Delete area:** A mark can be deleted by selecting it.
- **Export marks:** When the marking process is finished the user can export the marks currently shown over the image. If a *.txt* file with the same name as the image does not exist in the image's directory, the program creates it and saves the marks in the way specified in section 3.2.2. If the file does exist, the program overwrites it, saving the new marks instead of the old ones.

4.2.1 Implementation

When implementing the graphical interface, the two libraries used were Tkinter and Matplotlib. Tkinter was used to display the file system explorer window to select the image. To display the chosen image where the user marks the areas of interest, MatplotLib[22] was used. This library counts with several tools to display, move or zoom in the image, which perfectly cover the *Navigate Image* use-case.

To encapsulate the behaviour of a marked area, a class *Mark* was implemented by extending the Matplotlib's *Patch* class. This class counts with all the methods necessary to manage the position of the area in the image, change the visualisation mode, and convert the concrete pixel positions to values normalised between 0 and 1, as needed by the dataset specification.

Since all the inputs for managing the marks (create, move and delete) will be performed via clicks, the State Pattern (figure 4.3, page 48) was implemented to alter the behaviour of a click depending on the state of the program. Different states were implemented for the following actions: create a fixed size mark, create a variable size mark, move and delete marks. The user can change states by using the number keys 1 to 4, the current state name is shown above the image.

All of this can be better seen in the program's class diagram, represented in figure 4.4 (page 48). The execution of the program follows very closely the order depicted in the use-case diagram. It starts by creating a Tkinter FileDialog that allows the user to select an image file. Then an ImageMarker object is created that will open said image as a Matplotlib Figure. Next, the system tries to open the *.txt* file that describes the areas marked in the picture. If

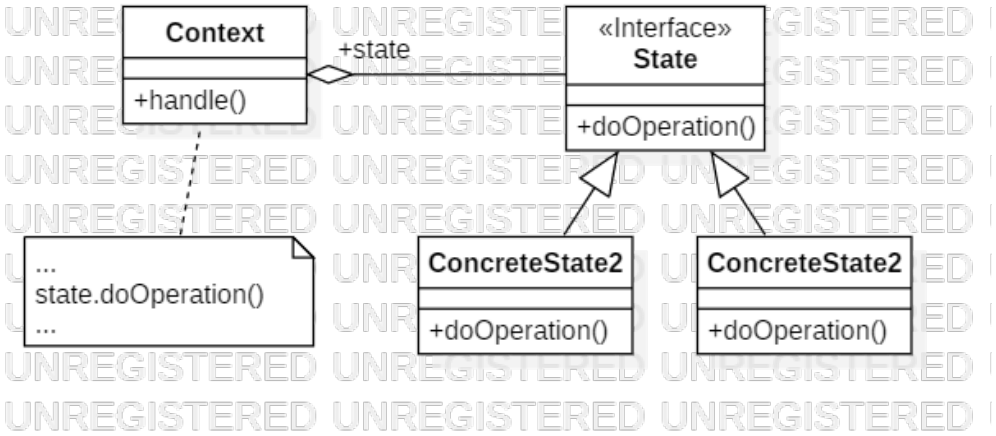

Figure 4.3: State Pattern class diagram

this file exist, an array of Mark objects is created from the normalised values present in the file by using the static method *buildFromNorm* from the Mark class. Then the main loop of the program begins, where the user can navigate the image and add, modify or delete marks. All of the inputs given by the user are processed by the ImageMarker object. The click inputs are directly passed to the current state of the object, while key inputs are processed by the object itself, allowing to change state or export the current marks.

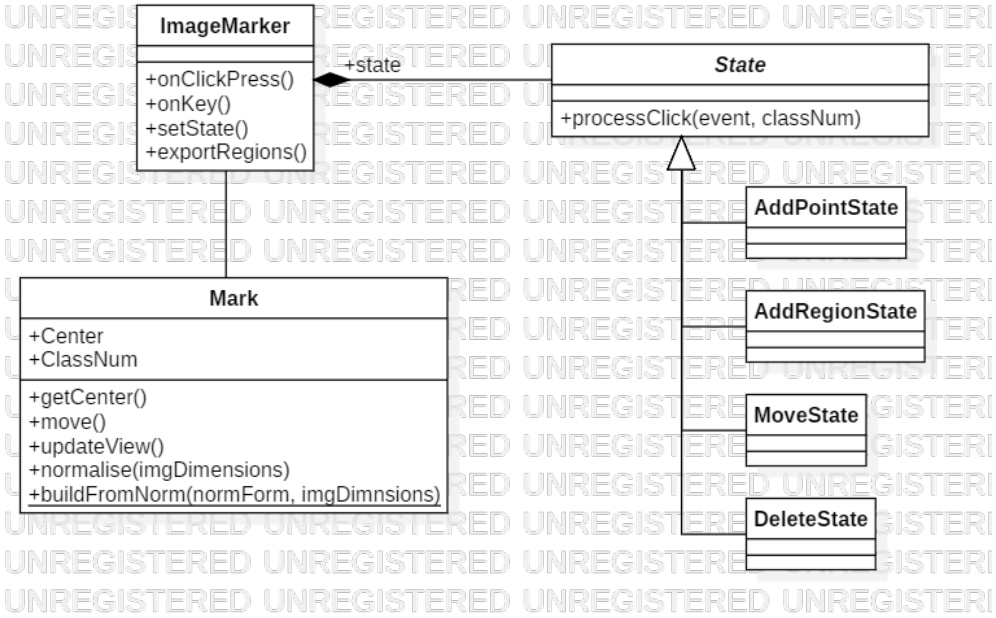

Figure 4.4: Image Marker class diagram

The development of a tool to annotate image datasets following the specification explained

in section 3.2.2 for their use in YOLO systems could constitute a project by itself. Since this is not the scope of this project, the program counts with several limitations in order to reduce its development time. As the problem requires just one type of class (a haemorrhage area) the program does not count with any mechanism to switch between different classes of objects being marked. Therefore, all the marks will be saved with class index 0, which in this context corresponds to a haemorrhage area. Also, no GUI or usability studies were performed when designing the program. The developer used most of the libraries functions without much customisation, in order to devote more time to experimenting with the DL system.

The following image (Figure 4.5, page 51), depicts the user interface of the program once a picture has been selected. The toolbar on the upper-left corner allows the user to navigate the image, and the blue boxes scattered across the image are the objects already marked.

Even though this program is in a basic pre-alfa state and much more work is needed for it to be a fully functional tool, it serves as a good starting point for further development. The classes were designed with extension in mind, allowing for simple upgrade paths in future developments. For example, if the system needs to be updated to support several classes, the Mark class does not need to be changed, just the ImageMarker needs to implement an attribute *currentClass* and pass it to the *State* when processing a click, since the state already creates a Mark assigning it the class number that is passed as a parameter.

Regarding the scope of this project, the program's functionality was enough to accomplish the objective of annotating the dataset in a comfortable manner. It is also clear that, with the need to enhance the dataset will come the need for upgrading the tool's ease of use and user friendliness.

4.2.2 User guide

This section serves as a brief user guide of the Image Marker program, detailing its control scheme and its intended use. First of all, the key-bindings and behaviour of the different states is shown in table 4.3 (page 50

The intended use of the program is the following:

- **Open an image:** Upon execution, a dialog will pop-up asking the user to choose an image. The user may then navigate, using the GUI, towards the image that needs to be marked. Figure 4.6 (page 52) shows said dialog box

- **Navigate the image:** The image is then opened, loading previously saved marks if possible. The user then uses the controls to create, move or delete marks in different parts of the image. The ”move” state is recomended when zooming or moving around the picture to avoid the accidental creation or deletion of marks.

Key	Action	Behaviour
º	Changes the visualisation mode	Alters between showing the bounding boxes or the centre point of the marks
1	Change state to add a fixed size mark	When the user clicks a spot in the image a $7x5px$ mark will be created centred on the clicked pixel
2	Change state to add a variable size region	The user will click first on the top-left corner of the region that is to be added and then on the lower-right corner to finalise the procedure.
3	Change state to move	This state serves two functions. First, it allows the user to click the picture without setting any mark. This is used to use Matplotlib's image navigation tools. And secondly, it allows the user to click and drag a region to change its position.
4	Change state to delete	When clicking inside a mark, it will be deleted
0	Export marks	All the marks present in the picture will be exported to a *.txt* file in the same directory and with the same name as the picture. If it does not exist, the file it is created. If it does exist, it is overwritten.

Table 4.3: Image Marker key-bindings

- **Export marks:** The user presses the ”Export marks” button to save the current marks in a *.txt* file in the same directory as the image. This file will loaded if its designated image is selected in subsequent executions of the program.

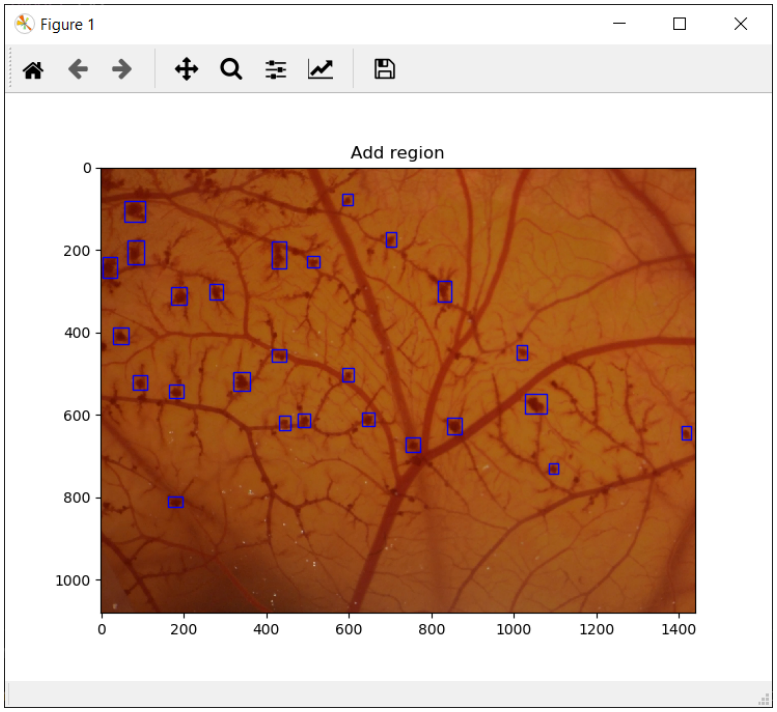

Figure 4.5: User interface of the Image Marker script

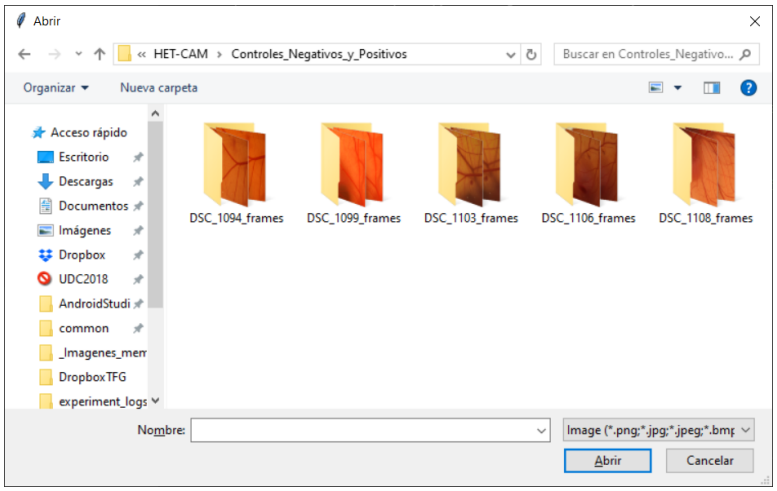

Figure 4.6: Open file dialog window

Chapter 5

Results

This chapter is devoted to present the results provided by the proposed methodology detailed in the previous chapter. On one hand, the outputs provided by the bleeding detection module will be analysed at different points throughout several positive HET-CAM videos. On the other hand, the information generated by the bleeding evolution module will be assessed regarding the distinction between negative and positive videos, as well as the relation with the experts' timings. Finally, the extracted results will be discussed highlighting the potential and constraints of the proposed methodology.

5.1 Detection of bleeding in HET-CAM videos

From the five models presented in section 3.3.9, the model corresponding to the best epoch of fold 4 was picked to test the analysis capability of the network. This concrete model was chosen because its performance is the closest to the average metrics calculated over all folds. This model is expected to represent more accurately the average performance of the network.

This section will tackle the qualitative results of the network on a frame by frame basis. In order to know if the evolution of haemorrhage is being calculated properly, a visual inspection of different video frames was undertaken.

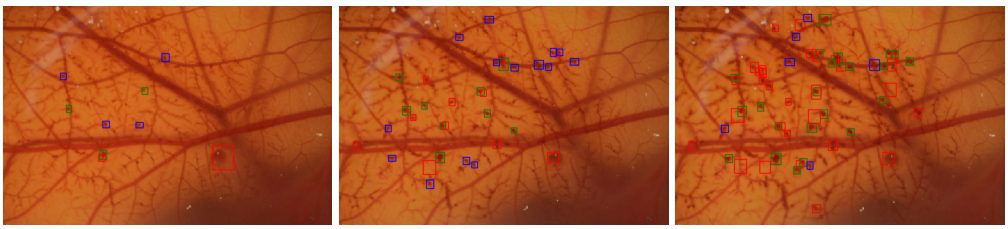

Figure 5.1: Detection in three equispaced frames in video DSC_1104

Figures 5.1 (page 53) and 5.2 (page 54) show three frames separated by a 240 frame margin in the beginning of the DSC_1104 and DSC_1107 videos respectively. At a first glance, the

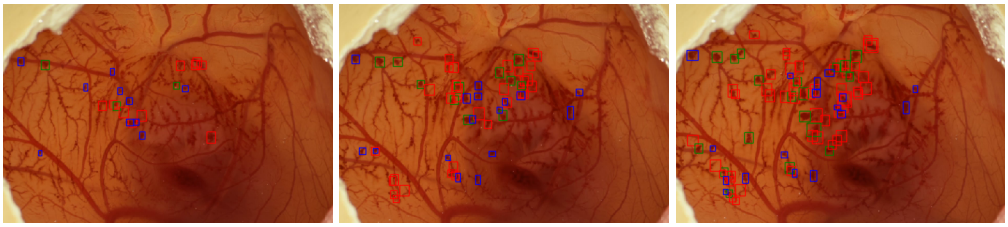

Figure 5.2: Detection in three equispaced frames in video DSC_1107

red colour, belonging to the false positive detections, is what catches the eye, but upon closer inspection one can see that a big part of these detections correspond to either non-annotated haemorrhages or do overlap with the ground truth but they do not reach the minimum 0.5 IoU. Nonetheless, it can be observed that the evolution of the overall number and size of detections does correspond to the growth of haemorrhage spots of the images, therefore allowing to conclude that the trained network's detections correlate in a consistent manner to the amount of bleeding present in each frame, thus allowing to build reliable graphs that show the evolution of such areas along the video.

One limitation to this affirmation comes from the process of tagging the dataset. One of the rules followed to mark haemorrhage spots was not to target blurry, low contrast, haemorrhage points, therefore, towards the end of the videos where these kinds of haemorrhages begin to appear, the model does not detect them. This phenomenon, observable in figure 5.3 (page 54), makes it so the amount of blood detected begins to decrease when the haemorrhages (detected in previous frames) begin to blur to the point where they are no longer detected.

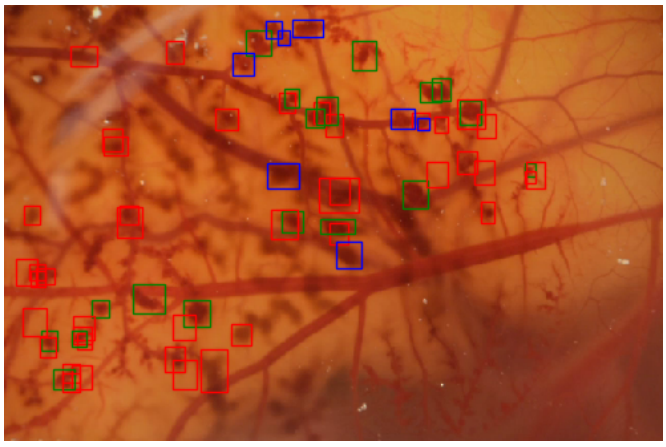

Figure 5.3: Blurry haemorrhages not detected towards the end of DSC_1104

5.2 Evolution of bleeding in HET-CAM videos

With the information gathered in the previous section, it is certain that the network does gives an accurate measurement of the variance of bleeding area between consecutive frames. If this concept is extrapolated to the whole video, the evolution of the bleeding area graph can be printed and their results analysed. As it is observed in figure 5.4 (page 55), the positive videos (in red) are clearly distinguishable from the negative ones (in blue). This was the expected result due to the testing performed over negative frames. Even if the graph belonging to negative videos is not completely flat, the tiny amount of detections that occur in some frames do not qualify the video as a candidate for being positive. Therefore, it can be concluded that the network clearly distinguishes between negative and positive videos.

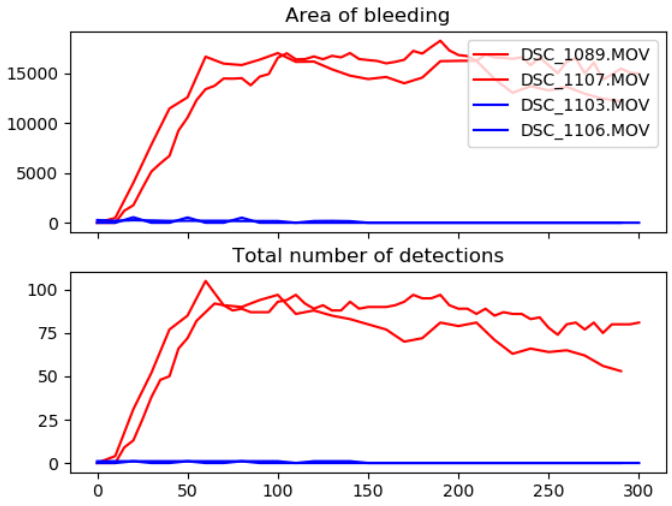

Figure 5.4: Evolution of haemorrhage in positive and negative videos

Comparing the output of the system with the timings given by the experts is more complicated matter. The times recorded by the experts on the videos of the dataset are stated in table 5.1. In each cell there is an array of values corresponding to the time given by each of the experts, always in the same order.

Just to remind, haemorrhage, lysis and coagulation are the three stages which timings need to be studied to calculate the irritation score of a drug. Haemorrhage marks the first appearances of bleeding spots, lysis occurs when the blood vessels are at their most dilated state and break, and coagulation marks the solidification of the blood that has spilled from the vessels.

One thing that these times have in common is that, on most of the videos, all of the stages are marked in less than a minute. This lessens the problem stated in the previous section,

Video	Haemorrhage	Lysis	Coagulation
DSC_1089	2, —, 2, 1, 1	8, —, 6, 5, 5	42, —, 7, 24, 18
DSC_1098	8, 9, 4, 3, 3	10, 18, 6, 5, 8	16, 74, 12, 21, 15
DSC_1104	1, 4, 2, 1, 1	3, 41, 4, 2, 6	20, 81, 5, 8, 12
DSC_1107	2, 10, 2, 2, 1	7, 59, 5, 4, 9	24, 91, 9, 14, 13
DSC_1109	1, 2, 1, 1, 1	2, 10, 2, 2, 2	20, 34, 4, 12, 6

Table 5.1: Experts' times (in seconds) on dataset positive videos

where the system loses reliability due to blurry haemorrhages towards the end of the video.

From taking a look at the experts' timings, a lot of variability can be seen. For example, in video DSC_1107, expert number 3 calls the coagulation time even before expert number 2 calls the haemorrhage time. This is one of the main goals of this project, to provide the experts with more objective data before making a decision in order to avoid these large discrepancies.

Some of the annotated videos were passed through the network in order to find a relation between the output of the system and the timings of the experts. From the first series of videos, whose outputs' are depicted in figure 5.5 (page 59), it was found that the system does not have the accuracy necessary to relate the output with the experts' times, since usually haemorrhage and lysis occur in the first few seconds of the video where the network has not detected anything yet.

Not only that, but due to the large amount of variance observed in the times given by the experts, it is not viable to evaluate the model by comparing it with these values. As the separation between the three stages is tied to small, concrete, changes in the images, figure 5.5 (page 59) demonstrates that the network is not accurate enough to detect these small events. If the system was able to detect reliably the first small haemorrhage spot, the haemorrhage time could be tied to that discovery, but due to the low recall, in the model being studied this would be just a game of chance. Nonetheless, this does not render the graph unusable. As it was said before, this tool is proposed to aid the experts in categorising the toxicity of a drug, not replacing them.

It can be observed that the videos analysed, which correspond to the same substance, show similar slopes in their graphs.It is expected that the evolution of the bleeding areas will vary with the toxicity score of a solution. This could make a possible line of future work using the network to differentiate between a set of substances, but for that a much greater dataset would be needed. Nonetheless, the graph, even though it does not show the separation between the stages, gives an objective measurement of the amount of bleeding in a video that can be used by the experts to quickly distinguish between different amounts of toxicity at the glimpse of an eye. This can be useful while experimenting, for example, with different concentrations of a solution. In this case the exact toxicity score does not need to be calculated for each one, but an automated system that shows what is the critical concentration until the solution becomes

too irritant can be developed using this same approach.

5.3 Potential and constraints

It was found that the final model does perform a useful function in the analysis of HET-CAM videos. For one, it was observed that high performance metrics are not needed to adequately differentiating between positive and negative videos. The trained network is completely capable of doing this distinction, allowing for an automation of the HET-CAM process, where the experts just need to focus their attention in positive videos, because negative ones can be filtered automatically by the program.

Secondly, with respect to the graphs that show the evolution of bleeding areas through the video, they were found useful in analysing the toxicity of a drug. Due to the variability present in the experts' timings, no relation was found between these times and the output of the network. Nonetheless, the graphs correctly show the evolution of haemorrhages in the videos and as such, they can be used as a guide to help experts with the analysis process. These was the main objective of the project. A more direct correlation between the haemorrhage, lysis and coagulation times and the graphs would be considered a more desirable output, but the graphs do give objective and relevant information to the experts analysing the videos. For example the slope of the curves relate to the amount of haemorrhages and how rapidly they are growing, which is a measurement of the toxicity of a substance. With more data, a relation could arise between the slope and the toxicity score, or the family of the substance.

Even though the final results are satisfactory, the network is not without limitations. First of all, the amount of data available is directly related to the potential performance of the network. Deep Learning networks need a much larger amount of tagged data than is available to be properly trained and tested. The main advantage of this type of system resides on being able to automatically recognise the common set of features that all objects in the dataset share. A dataset can be considered as a subset of all the data available for a concrete context, for example, the HET-CAM test. The smaller it is, the more characteristics of that concrete subset will be learned by the network, and thus, the less generalistic the model obtained will be.

Regarding this specific project, the size of the dataset is limited in two fronts. First of all the amount of raw data available. A greater number of videos, under different lighting conditions or different types of irritant agents, would be desirable. There is a finite amount of data that can be extracted from one video. As it was said before, consecutive frames were not considered to be included in the dataset because of their similarity due to the slow rate of growth of haemorrhages, trying not to encourage the network to learn a concrete shape of haemorrhage. And even if the frames are spaced by a couple of seconds, the basic shape

of the haemorrhage tends to be similar, therefore, for each video, the dataset is gathering a number of haemorrhages and 10 possible variations of those bleeding spots. It is easy to see that the network would learn more from 10 frames belonging to completely different videos than from 10 frames of the same video (even if they are spaced apart from each other).

Apart from the amount of videos available, the marking process of the individual haemorrhage spots counts with its constraints. A tool was developed to help in this front, but targeting every single haemorrhage in every single frame is still tedious, time-consuming and not realistic in the time frame devoted to this task. Some automation was tried, like replicating the targets in the previous image and then shifting and resizing all of them to try and encapsulate the haemorrhages that were present in the previous frame, accounting for their growth. This was not effective since not all the haemorrhages are in the same vessel layer and the layers move at different rates. Therefore, each haemorrhage needed to be annotated individually thus reducing the amount of data that can be gathered in a fixed amount time.

To conclude, the system developed is considered a capable proof of concept that opens the way to several future lines of work. It was demonstrated that the distinction between positive and negative videos can be automated using an artificial intelligence system. Furthermore, the area of bleeding graphs described by the system do correlate with the expected evolution observed in the videos, having very distinct shapes that can lead to further automation by analysing the shape of the graph and relating it to a kind of substance or toxicity score. This project serves as reaffirmation that deep learning networks can help with the gathering of valuable information and help future standardisation of the HET-CAM test, and medical science in general, to supply a more stable basis for the experts to make studies upon.

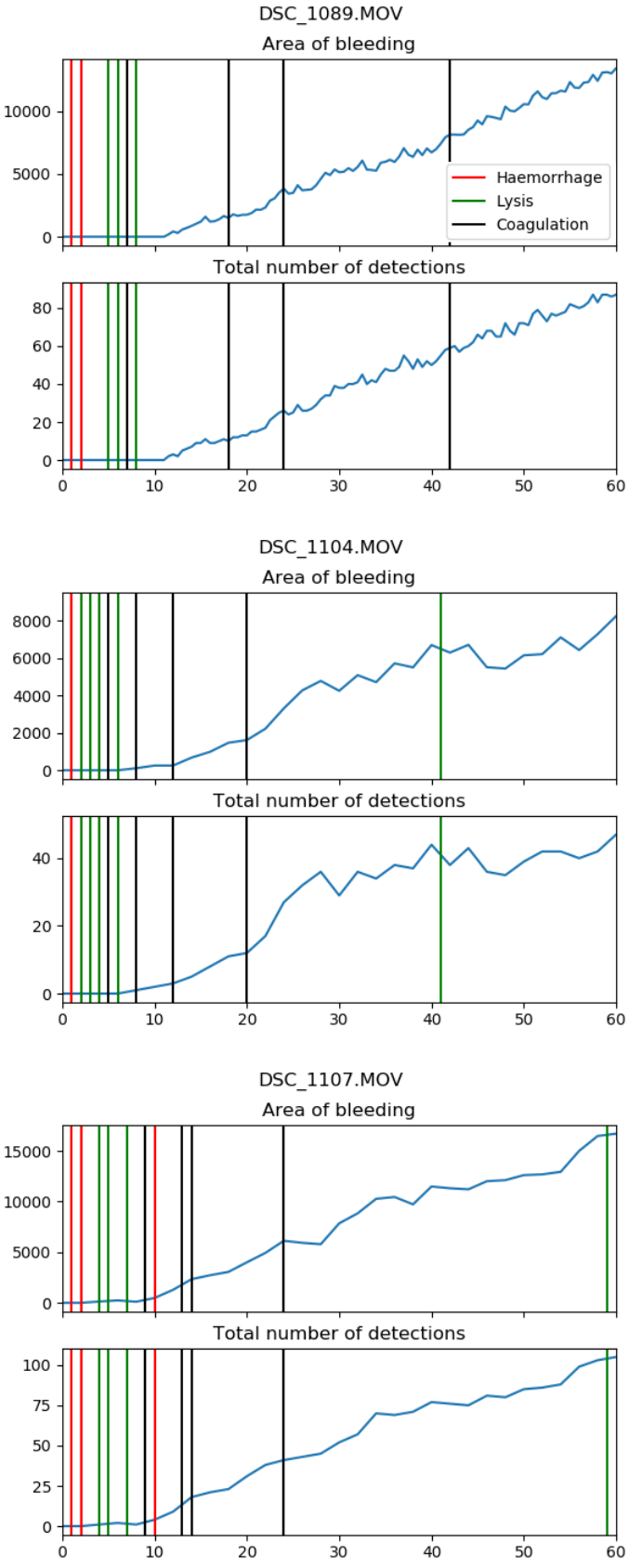

Figure 5.5: Timing of the experts in several analysed videos

Chapter 6

Conclusion and future work

This constitutes the final chapter of the project, where a retrospective of the problem, methodology and results will be presented along with future lines of work and enhancements that can be applied to the project.

6.1 Conclusion

Throughout the document, the different parts that comprise this project were explained. The process began by explaining the motivation of the problem and acquiring a more knowledge of the context, from the HET-CAM test to object detection using deep learning. Then, the HET-CAM videos available were annotated to conform a dataset. For this purpose, several scripts were developed, from separating the videos into individual frames, to a GUI to help target haemorrhage spots in the selected set of frames. Then, the base YOLO network used in the project was fine tuned with regards to different parameters to optimise the performance of the final system. Some of the parameters experimented with where learning rate, data augmentation, or pretrained weights of the network. The experimentation process concluded with the selection of the final model that was used to analyse the videos. Lastly, a script that uses the model to analyse the videos regarding the amount of detected bleeding spots and haemorrhage area was developed. The videos available were passed to the script in order to analyse the outputs. It was found that the system can reliably recognise between positive and negative videos. Nonetheless, the timings given by the experts on the different stages of the videos were found to be vary sparse, therefore no relation between the output of the network and the times was found. In this section the conclusions gathered from the whole process are presented.

First of all, the relevance of this project needs to be outlined. In the last few years, a lot of awareness has been raised regarding animal rights and the need to stop unnecessary animal suffering. Therefore, an increase in the usage of procedures like HET-CAM instead

of in vivo testing is expected in the following years. With this in mind, it is of the utmost importance to refine and standardise the procedure, to make it easier for the industry to adopt it. The tool that was developed in this project serves as a step forward in this front. The use of artificial intelligence allows this program to be more generalistic than other, more algorithmic, approaches. This encourages the industry to use this kind of procedure, since it is resilient to ilumination, resolution or framerate changes, thus not making almost any restriction in the process of recording the video. Making the procedure more accessible can greatly increase the number of users that adopt this measure compared to other approaches.

One of the main motivations of this study is the need for a common ground on analysing the HET-CAM test. It was observed in the experts annotations that, as they are based on a subjective ocular inspection of the video, there is a lot of variance between them. This is mainly caused by the nature of the experiments, as it is difficult to visually isolate the concrete event that causes each phase of the process. The program can serve as a more objective form of data that, added to the video, can help the experts on determining the timings of each of the three stages studied for toxicity analysis. This tool is not destined to replace the role of the experts in HET-CAM analysis, but to provide an objective source of data (the haemorrhage evolution graph) so they make more informed decisions and reduce the subjectiveness of the process.

The final program is still a work in progress, and thus several improvements are needed in order to train a real-world-capable model. The trained model's performance is constrained in two main fronts. First of all, an appropriately-sized dataset is crucial in artificial intelligence problems. In this case, due to the amount of videos available and the time devoted to annotate them, the ideal size of the dataset is far from being reached. This prevents the model from being able to generalise the characteristics of a haemorrhage spot in order to detect them. The lack of videos caused the program to have a very low performance. Nonetheless, it was found that the system was able to easily recognise between positive and negative videos.

Regarding the correlation between the times of the experts and the output given by the network, no relation was found. The main cause of this is the amount of data available. With only 5 videos available, the distribution of the different experts' times is sparse enough to which no relation between them can be easily seen, specially with regard to the coagulation time. IT is expected that, as the size of the dataset grows, the variation between the times given by the experts will decrease, and the performance of the network will increase to a point at which a correlation can be found.

As a final note, the program developed is considered a good proof of concept of the standardisation of the HET-CAM procedure using deep learning techniques. Even though the final results are not ideal, the different tools, like the main program or the dataset annotation tool, were developed taking expansion into account and thus serving as good starting points

for future lines of work.

6.2 Future work

It is clear that the project portrayed in this document has a limited size and just serves as a pilot for future development. Much more work needs to be done in order to transform this program into an standard for analysing HET-CAM videos. This section explains the most important lines of future work towars achieving this goal.

First, and most importantly, a large amount of work still needs to be done regarding the size of the dataset. It is expected that more videos and annotated frames will have a hefty impact on the performance of the network. Of course, if a good amount of time is going to be devoted to enlarging the dataset, the first step to accomplish is to improve the dataset marking tool. Even though this build was sufficient for the scope of this project, an extended marking process would clearly benefit from some improvements in this front. Improving the usability as well as adding new functionality, such as opening a series of images or even reading video frames directly, would yield a substantial increase in efficiency. Since this is the biggest bottleneck that the system has, it is imperative that a good amount of the workforce devoted to future work will be assigned to this part.

If the dataset is expanded upon, the model would need to be trained over the new set. This could change the results of some experiments, like the learning rate or data augmentation, therefore this experiments would need to be redone. With this newly trained and optimised model, better performance can be achieved and in that case, the analysis of the graphs should be revisited. With a superior network, some other relation between the processes taking place in the videos and the graphs printed by the program can be found. Another advantage of counting with an extend dataset is that the haemorrhage, lysis and coagulation times, that seem a bit sparse in the 5 videos that are currently in the dataset, can be analysed in order to find a better average to compare against, or even allow to relate the output of the network with a concrete substance.

Regarding the final program, the enhancement of the UI can be considered as a way of, for example, encapsulating the most commonly altered parameters in an options menu, thus improving usability and experimentation, Another line of work can consist in allowing the analysis of a live video feed instead of having to pre-record the videos to analyse them. The network architecture used is known for being capable of working at nearly 60 frames per second, therefore analysing live video is within its working specifications. Not only that, but not all the videos available are shot at less than 60 frames per second, however all of them are considered valid HET-CAM tests. This points to a line of work on finding the lowest acceptable frame rate at which the videos can be properly analysed both by the system and

the experts. Reducing the recording frame rate would improve the time performance of the model in pre-recorded videos, allowing the network to process more tests in the same amount of time.

All of these improvements can be compiled into a more complex system that serves as even more of an aid to the experts analysing the toxicity of substances. The shape of the graphs corresponding to different substances' videos can be parametrised in order to apply a classification algorithm to categorise new samples in a certain substance family (or toxicity range). This, along with other changes like automatically detecting negative videos can lead to bast improvements in the classification of drugs by toxicity in the pharmaceutical industry.

Appendices

Appendix A

Project planning

This appendix explains the process behind the planning and scheduling of the project. The process begins with the division of the entire project in a set of tasks. The relations and dependencies between tasks are taken into account before assigning each one to a given time frame. Finally the tasks are scheduled in order to meet the deadline.

The whole project was divided in three stages which correspond to three chapters of this report: domain research, methodology, and, results analysis and report. Each of the these is subdivided into more granular tasks that need to be accomplished.

- **Domain research:** This stage corresponds to acquiring an understanding of the domain that is going to be the basis for the work developed in the project. Concretely, the two fields studied that are going to be studied are the HET-CAM process, and deep learning (more concretely the YOLO model). Not a lot of time was devoted to this stage due to the fact that the project developed by Gende [3] was a very valuable source of information with regards to the HET-CAM topic.

- **Methodology:** It is comprised of the marking of the dataset, the implementation of the YOLO network and experimentation in order to find the most suitable parameters. To finalise, the development of the final bleed area detection module was performed. This stage encompasses most of the work that needs to be done in the project and thus it is going to be allocated the most time.

- **Result analysis and report** Finally, the module developed is going to be put to the test by analysing both negative and positive videos in order to check if the output of the program is appropriate. This will constitute the final stage of the project and will end with the writing and refining of this report.

Allocating a certain amount of time to a task is one of the key components of good project planning, but it is also one of the most difficult. Due to the lack of experience of the developer,

the planning is expected to vary due to the lack of precision in assigning a task's time frame. Nonetheless, the initial schedule is shown in figure A.1 (page 68).

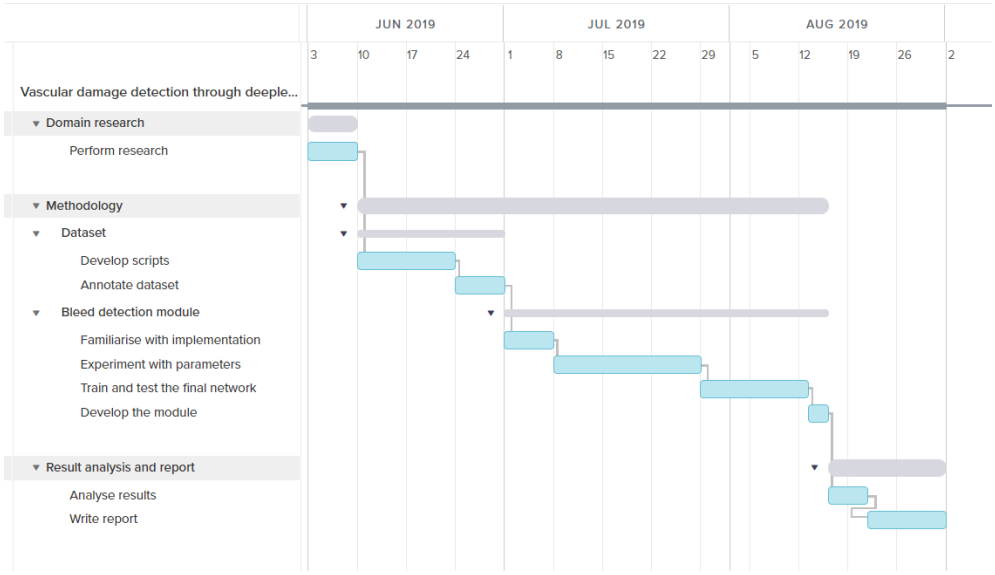

Figure A.1: Initial project planning schedule, divided into tasks

When following this schedule, several unexpected problems were encountered that increased greatly the amount of time that needed to be allocated to some tasks. Mainly the task corresponding to the experimentation performed over the network. This task was widely underestimated, the complexity of the network and the time devoted to analyse each experiment's results was greater than expected. Also some experiments need to be redone, and network took 6 to 8 hours to perform a whole cross-validation process.

Due to these unprecedented changes, the deadline of the project needed to be pushed back from September to November. The final state of the schedule can be seen in figure A.2 (page 68).

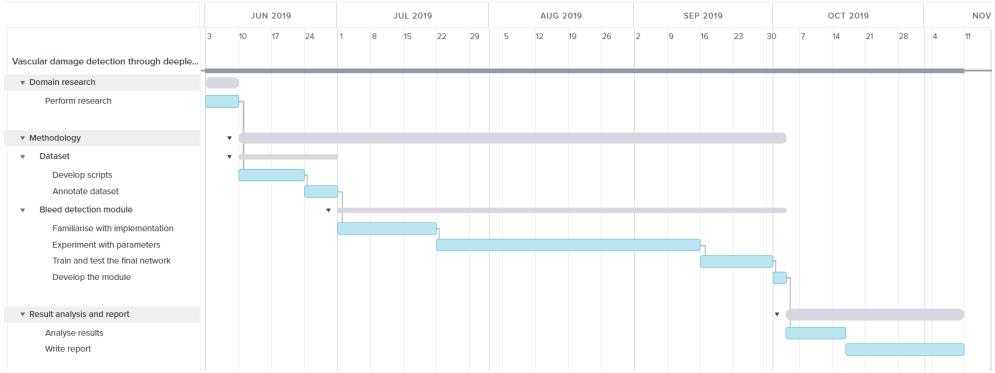

Figure A.2: Final project planning schedule, divided into tasks

List of Acronyms

AI *Artificial Intelligence.*

CNN *Convolutional Neural Network.*

DL *Deep Learning.*

FP *False Positive.*

FN *False Negative.*

HET-CAM *Hen's Egg Test - ChorioAllantoic Membrane.*

IoU *Intersection over Union.*

IS *Irritation Score.*

IT *Information Technologies.*

TP *True Positive.*

YOLO *You Only Look Once.*

Glossary

Classification Problem that consists in categorising a sample into a finite number of classes.

Regression Problem that consist in categorising a sample into an infinite range of values (i.e a percentage).

Aspect ratio The relation between width and height of an image.

Dataset A collection of annotated data relating to a certain context

Training set Set of data used in the training process of an AI system.

Validation set Set of data used to iteratively evaluate the performance of an AI system during the training process

Test set Set of data used to check the performance of a trained AI system

Bibliography

[1] K. R. Wilhelmus, "The draize eye test," *Survey of Ophthalmology*, vol. 45, no. 6, pp. 493 – 515, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0039625701002119

[2] N. Luepke and F. Kemper, "The het-cam test: An alternative to the draize eye test," *Food and Chemical Toxicology*, vol. 24, no. 6, pp. 495 – 496, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0278691586900992

[3] M. G. Lozano, "Caracterización de la toxicidad en membranas mediante técnicas de procesamiento digital de imagen," Bachelor's dissertation, Universidade da Coruña, September 2018.

[4] P. Budai, J. Lehel, J. Tavaszi, and E. Kormos, "Het-cam test for determining the possible eye irritancy of pesticides," *Acta Veterinaria Hungarica*, vol. 58, no. 3, pp. 369–377, 2010.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 5 2015.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[9] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *in Proc. Int. Conf. Acoust., Speech, Signal Process*, 2013.

[10] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[11] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[12] R. Girshick, "Fast r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[13] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[14] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[15] "Simultaneous detection and classification of breast masses in digital mammograms via a deep learning yolo-based cad system," *Computer Methods and Programs in Biomedicine*, vol. 157, pp. 85 – 94, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169260717314980

[16] S. R. S., J. George, S. Skaria, and V. V. V., "Using YOLO based deep learning network for real time detection and localization of lung nodules from low dose CT scans," in *Medical Imaging 2018: Computer-Aided Diagnosis*, N. Petrick and K. Mori, Eds., vol. 10575, International Society for Optics and Photonics. SPIE, 2018, pp. 347 – 355. [Online]. Available: https://doi.org/10.1117/12.2293699

[17] Y. LeCun, "Mnist handwritten digit database, yann lecun, corinna cortes and chris burges," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[18] ——, "Imagenet," 2016. [Online]. Available: http://www.image-net.org/

[19] E. Linder-Norén, "Pytorch-yolov3," 2019. [Online]. Available: https://github.com/eriklindernoren/PyTorch-YOLOv3

[20] J. Redmon, "Yolo: Real-time object deteection," 2018. [Online]. Available: https://pjreddie.com/darknet/yolo/

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[22] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[23] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.